The HP-IL
Interface Specification

HEWLETT PACKARD

The HP-IL Interface Specification

November 1982

82166-90017

Preface

Chapter one provides an overview of the HP-IL interface system with some example implementations. The complete functional, electrical, and mechanical specifications of HP-IL are contained in chapters two, three, and four respectively. Chapter five discusses system considerations such as recommended sequences of interface messages, asynchronous loop operations, and device compatibility. In the appendices information such as message coding and glossary of terms may be found. This manual is not intended to be a tutorial on HP-IL or I/O techniques, therefore, familiarity with other interfaces is very helpful. Because of the extensive similarity between HP-IL and HP-IB, an understanding of the latter is recommended, though not required.

CONTENTS

Chapter 1:   Introduction

Chapter 2:   Functional Specifications

Chapter 3:   Electrical Specifications

1. INTRODUCTION

1.1 General Information

HP-IL (Hewlett-Packard Interface Loop) is a system which permits communication between devices. In comparison with other interface systems HP-IL is small, low power, low cost, and medium distance. HP-IL has the capability to transfer data at a faster rate than commonly available serial interfaces. As the name implies, devices are connected in a circular loop structure. Digital messages travel from one device to the next around the loop in one direction only. All devices must obey certain functional, electrical, and mechanical rules in order to communicate by means of HP-IL.

This chapter provides an introduction to the HP-IL interface specifications which are presented in chapters 2, 3, and 4. Because this manual is not tutorial in nature, the novice may wish to refer to "THE HP-IL SYSTEM"* for a complete tutorial before proceeding.

1.1.1 Application Examples

The following three applications help show the wide range of capabilities available with HP-IL.

For his dissertation, a forestry graduate student is studying the effect of various types of herbicides on underbrush, tree growth, etc. Observations are made on several different plots many miles from the school. Because of its low cost, small size, and continuous memory feature he uses the HP-41CV handheld computer as a data collector. A short program he has written prompts him for the correct data items and stores them in the appropriate registers. The university forestry lab has a small HP-IL system including the HP 82160A HP-IL Module for the

* Gerry Kane, Steve Harper, and David Ushijima, "THE HP-IL SYSTEM: An Introductory Guide to the Hewlett-Packard Interface Loop" (Osborne/McGraw-Hill 1982).

HP-41CV, the HP 82161A Digital Cassette Drive, the HP 82162A
Thermal Printer, and the HP 7470A Plotter. When the student
returns the lab he connects his HP-41CV to the HP-IL system and
inserts a minicassette into the tape drive. The cassette
contains his programs and the data he has collected on his
project over the last several months. He loads and runs a
program from the minicassette which reads the data taken today
from the HP-41CV and stores it on the tape with the rest of the
data. He then loads another program which uses all the collected
data on the tape to make finished graphic plots and histograms
of various combinations of the data, such as toxicity levels over
time, amount of undesirable foliage with respect to the type of
herbicide, etc. Prior to the HP-41CV and HP-IL, the cost of such
an application would have been prohibitive. Furthermore, the low
power requirements and the power down feature of HP-IL permit
unattended operation for extended time periods. If the student
had needed to measure toxicity level, for example, every hour for
a week, a small system in a weatherproof box could be left on
site under battery power. Naturally, the measuring apparatus
would need to be connected to the HP-IL system, but this could be
accomplished with the HP 82166A HP-IL Converter.

     Each jewelery store in a small chain uses from two to five
point-of-sale terminals consisting of an HP-75C computer and
an HP 82162A Thermal Printer. These devices are all connected on
HP-IL to an HP 82161A Digital Cassette Drive and an HP 82905B
80 column printer located in the back room of the store. Each
HP-75C contains a program which prompts the salesclerk for all
necessary information about each transaction. The data, which
might include item description, inventory number, price, and
charge card number is stored on the cassette drive. A customer
receipt is printed on the small printer on the counter and a
transaction summary is printed simultaneously on the larger
printer in the back room. At any time, a special program can be
used to generate sales reports, lists of items to be re-ordered,
etc. With the HP 82164A HP-IL/RS-232-C Interface and a modem the
transaction or inventory data could be transmitted by telephone
to the main store. Prior to HP-IL, such a system would have been
much too expensive for very small businesses.

     A small electronics firm is about to begin volume
production of a new audio amplifier printed circuit board. The
production engineer has designed an automatic test system
consisting of the HP-85A Personal Computer with the HP 82938A
HP-IL Interface, and the HP 3468A HP-IL Digital Multimeter. The
engineer has also built a special device containing a test
fixture for the PC boards, a programmable power supply, a
programmable waveform generator, and relay switches to connect

these devices to the various PC board test points. This special
device uses the HP 82166A HP-IL Converter so that it too is
controlled with HP-IL. After initial debugging, the engineer
writes the test program so that up to three identical test
stations can be controlled on the same loop since he has found
that HP-IL and the HP-85A are fast enough to support this
throughput. Test programs and results are stored on the internal
cartridge tape drive for future failure analysis. Later when a
second version of the amplifier board goes into production some
simple program modifications permit testing of both versions at
the same time on different test stations using the same system.
The low cost and flexibility of HP-IL were the deciding factors
in the use of automated testing as opposed to a traditional
manual approach.

1.2 Functional Overview

HP-IL is a master-slave interface system. One of the
devices on the loop is designated the loop controller and this
device has the responsibility to transmit all commands to other
devices on the loop. The HP-41C and the HP-85A are examples of
devices that can be HP-IL controllers when they are equipped with
the proper plug-in module. A device with the ability to send
device dependent data to other devices on the loop is called a
talker. Note that even though a device has the talker capability
it must not actually send its data until commanded to do so by
the loop controller. The HP 82161A Digital Cassette Drive is a
device which can be a talker. Listeners are devices with the
capability to receive data from the loop. Once again, listeners
must remain inactive until they receive a command from the
controller which enables them to receive data. The HP 82162A
Thermal Printer is a device which can be a listener.

Controller, talker, and listener functions are the three
basic capabilities of HP-IL devices. A device may have only one
of the three or may include some combination of capabilities as
is more often the case. Talkers often have listener capability
and controllers almost always have both talker and listener
capabilities as well. In general, every HP-IL system has all
three capabilities somewhere within its devices; however, under
special circumstances a system can be constructed with only a
talker and one or more listeners without the controller. In this
case, all devices must be able to operate in the talk-only or
listen-only mode. An example might be a voltmeter logging

readings on a printer. In larger loop systems there are usually
several talkers and listeners. The controller will permit only
one talker to be active at a given time, but may enable multiple
listeners to receive the talker's data, if desired. There may
even be several controllers on one loop. In this case, one of
the controllers (the system controller) is in charge when the
system is first turned on. Protocol exists within HP-IL to allow
the various controllers to take turns controlling the other
devices on the loop. Only one of the controllers is active at
one time.

     In addition to the talker, listener, and controller
functions the following capabilities are available on HP-IL:

- Devices may indicate a need for service from the
  controller using two different methods.

- Power down mode may be used to conserve power.

- The controller may assign addresses in three different
  ways for a maximum of 31 or 961 devices.

- Devices may be identified by model number or a device
  capability code.

- Devices can be triggered, cleared and commanded to
  ignore local controls and only respond to loop
  messages.

- The talker may be temporarily interrupted during long
  data transmissions so that other tasks may be
  performed.

     This wide range of capabilities permits the controller to
be in charge of both the configuration and the operation of the
loop. It is possible to implement high-level functions in the
controller which make the details of loop protocol entirely
transparent to the user. In such a friendly system, the user
would simply execute a function and the computer would
automatically configure the loop, find the desired device, and
send the appropriate commands and data.

There are two major classes of messages which are sent
over the loop. Interface messages are usually sent by the active
controller and are used to direct the operation of the system.
The controller's commands are an example of this type of message.
In general, these messages are considered overhead. The primary
purpose of HP-IL, of course, is the transmission of data. Data
messages are used as a communication medium and do not directly
affect the operation of the interface system. Data messages are
sent by the active talker and received by one or more active
listeners.

Normally only one message is in transit around the loop
at any given time. In general, when a device sources a message
it waits for the message to go completely around the loop and
return before transmitting the next message. Most messages are
held at each loop device until it is ready to accept the next
message. This is the normal loop handshake and it guarantees
that talkers and controllers do not send messages faster than
devices can accept them. This handshake also provides an
excellent error-checking capability. If the message returns the
same as it was sent, the sending device knows that all devices
must have received it correctly.

Clearly, if there are a number of slow devices on the
loop that delay each message until they are ready to pass it on
to the next device, the loop speed will be unacceptably low.
Fortunately, data messages are usually intended for only one or
two devices at a time. Other devices on the loop are in their
inactive state. Devices are built so that messages not meant for
them are passed on very rapidly. In this manner the loop
throughput can be maintained at an acceptable level.

Since commands are often meant for all devices on the
loop, the handshake process for commands is modified slightly in
order to maintain reasonable throughput. When a device receives
a command message, it passes it on immmediately but also retains
a copy of the message for interpretation. In this manner all
 devices can be executing the same command in parallel. The
message returns to the controller fairly rapidly, but its return
does not indicate that the devices are ready. The controller
must now send a special message called Ready For Command (RFC).
This message is held by each device until it is ready for another
message. If all devices execute the command in about the same
amount of time, there will be a delay until the first device is
finished and then the RFC message will move fairly rapidly around
the loop to the controller. Now the controller knows that all
devices are ready and the next message can be sent.
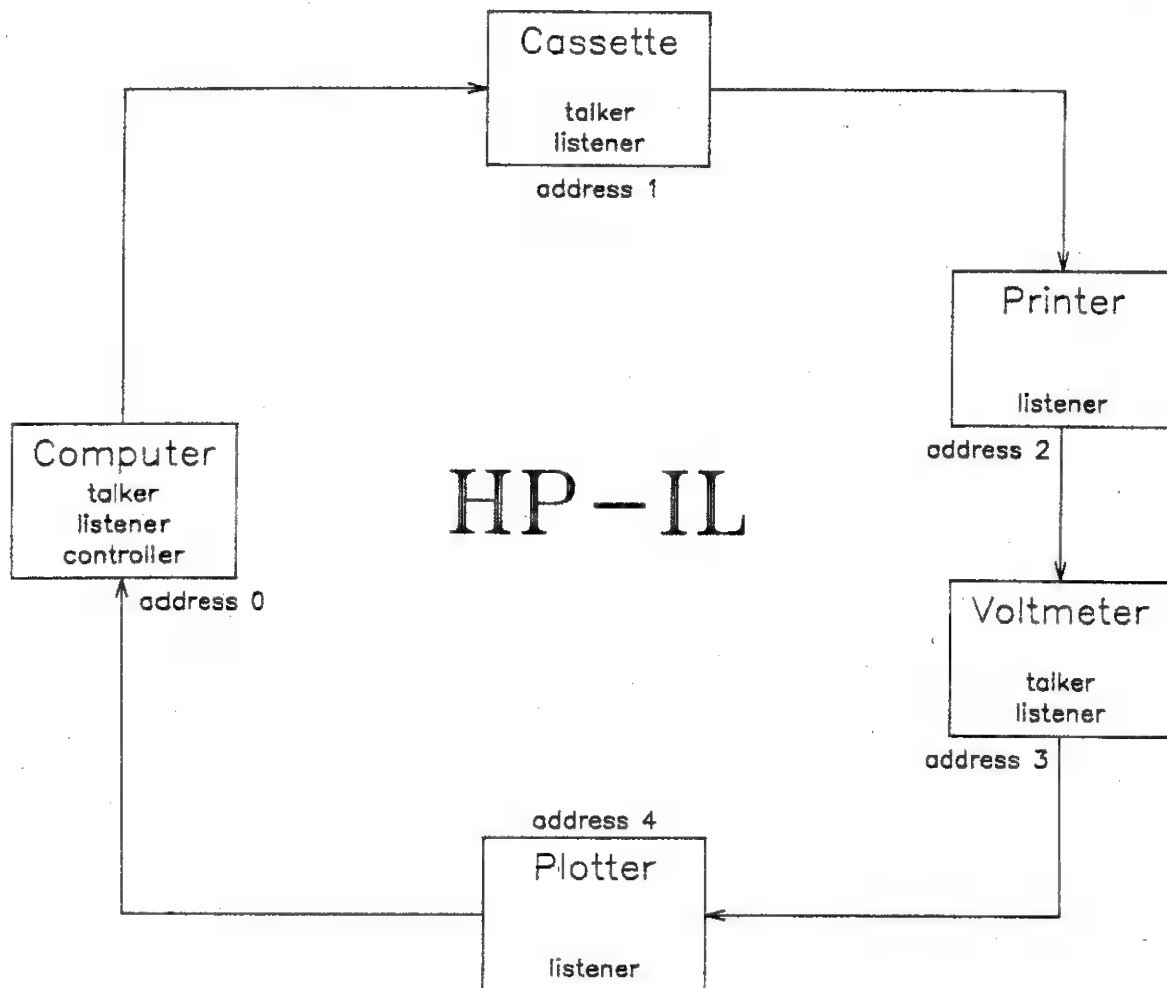
Figure 1-1.   Sample HP-IL System

          Suppose that we have an HP-IL system consisting of
devices with basic capabilities and loop addresses as shown in
figure 1-1.   Assume that the voltmeter is ready to transmit a
reading on the loop, and the user wishes to print it on the
printer.   The computer first sends the Unlisten command which
causes any previous active listener devices to become inactive.
When the command returns the controller sends the Ready For
Command (RFC) message as is required after every command.   The
calculator now sends the Talk Address 3 command (and the RFC) to
direct the voltmeter to be the talker and to cause any previously
addressed talker to become inactive.   At this point, even though
the voltmeter is addressed to talk, it may not send its data
until specifically instructed to do so.   The computer then sends
the Listen Address 2 command and RFC.   The printer alone
recognizes this command and becomes ready to receive data
messages.   This command does not cause other listeners to go
inactive, hence the need for the Unlisten command at the
beginning.   At this point the computer sends a special message

called Send Data to cause the voltmeter to begin sourcing data
messages. The Send Data is in the ready class of messages which
is used for handshake purposes. When the voltmeter receives the
Send Data it does not retransmit it, but instead sends its first
data message. The voltmeter sends a sequence of data bytes, for
example, +2.658VDC followed by carriage return and line feed
characters. Each data byte goes around the loop to the printer,
which holds the byte momentarily until it is loaded into the
buffer, and back to the voltmeter. After each byte is received
and error-checked the voltmeter sends the next character. When
the line feed returns and error-checks properly the voltmeter
sources a special End Of Transmission ready message. This
message signals the computer that the transmission is now
complete and that the computer should resume active control of
the loop. Similar to the Send Data message, the computer does
not retransmit the End Of Transmission but replaces it with the
next command message.

It must be noted that HP-IL protocol is based very
strongly on HP-IB (Hewlett-Packard's implementation of IEEE-488).
While it appears that HP-IL is functionally a bit-serial version
of HP-IB, the user is warned that there are some significant
differences in protocol. While familiarity with HP-IB is very
helpful, it is not necessary. All information needed to learn
and use HP-IL is contained in this document.

1.3 Electrical Overview

Every message on the loop is sent as a sequence of eleven
bits called a message frame. The first bit, called the sync bit,
is coded in a special way so that each device can easily
recognize the beginning of a frame. The sync bit and the next
two bits are called control bits and they determine the major
classification of a message. Command messages, ready messages,
and data messages are examples of these major classes. The
remaining eight bits, sometimes called data bits (not to be
confused with the data message) specify the particular message
within the classification. The Unlisten command frame, for
example, would be 100 0011111, while the Interface Clear command
is 100 10010000. The Send Data ready message is 101 01100000.
The space between the control and data bits is for clarity only
and does not represent a time delay or any other delimiter.

        The electrical connection from one device to the next is
a differential, voltage-mode, two-wire, balanced line. Both
wires are floating with respect to both devices' ground
connections. One of the wires is chosen as a reference and the
voltage of the other wire is measured only with respect to the
reference. This has several advantages. Device grounds need not
all be at the same potential. In fact there can be rather large
differences with no effect. This is especially handy for
voltmeters which need to measure values not referenced to ground.
It totally avoids the problem of ground loops in an interface
system. Since noise pulses tend to affect both wires equally,
the balanced nature of HP-IL strongly rejects these pulses. The
same holds true for noise radiation from the system itself. As
one wire's voltage rises, the other's falls, tending to cancel
any radiated signal.

        Bits are encoded using a three level, or bipolar code.
The voltage difference between the two wires may be nominally
-1.5 Volts, 0 Volts, or +1.5 Volts. A logical one is encoded as
a high pulse (+1.5 Volts) followed by a low pulse (-1.5 Volts).
A logical zero is a low followed by a high. A logical one sync
is a high, low, high, low sequence. A zero sync is low, high,
low, high. The nominal pulse width is one microsecond and each
bit sequence is always followed by a minimum delay time (0 Volts)
of about two microseconds. Refer to figure 1-2. This type of
code provides good noise immunity, is relatively insensitive to
speed variations, is self-clocking, and has no DC component in
the signal.
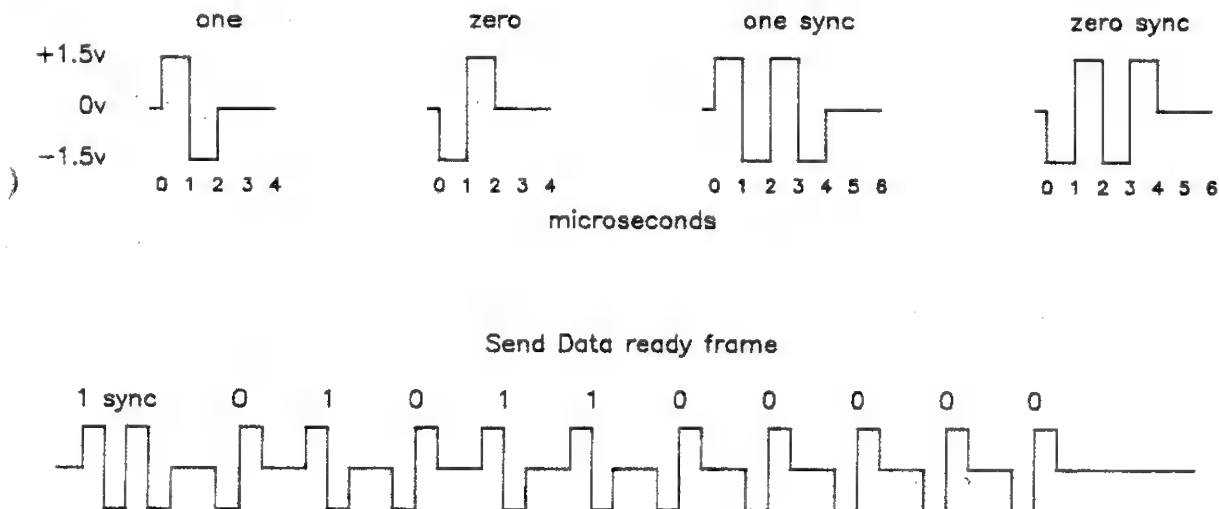
Figure 1-2.   HP-IL Bit Encoding

Frames are asynchronous with respect to each other. That means no common system clock is necessary in HP-IL. Furthermore, even bits within frames are asynchronous. The sync bit requires six microseconds to transmit while other bits take four microseconds. A complete frame, which may contain one byte of data, takes 46 microseconds assuming no extra delay between bits. If we further assume no extra delay between frames, a maximum loop rate of approximately 20 kilobytes per second could be achieved. In a typical implementation with present electronics, other hardware and software delays lower this rate to somewhere between three and five kilobytes per second. This is enough to transmit more than a full single-spaced typewritten page of data every second.

Figure 1-3 is a simple block diagram of the interface electronics used in a particular implementation of HP-IL in a device. The HP-IL input and output lines are isolated through simple pulse transformers. The discrete components provide the proper load value for input and filtering for EMI reduction with impedance matching for the output side.

Figure 1-3.   Example Device Electronics
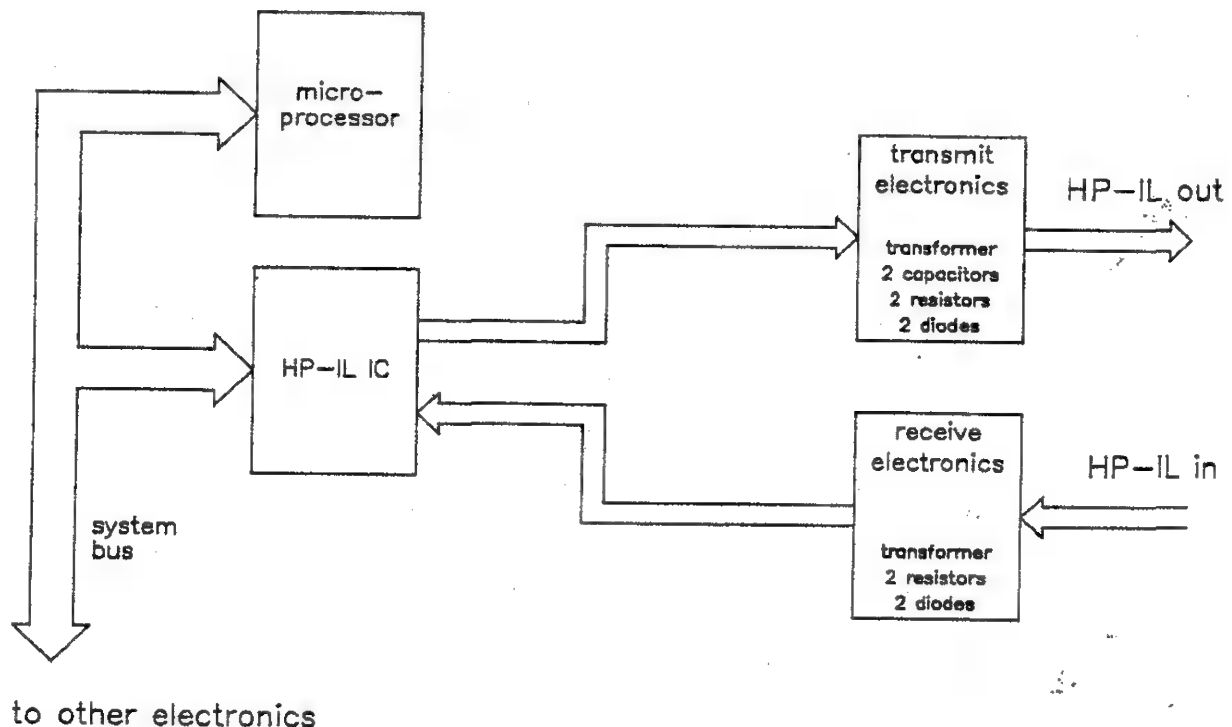
Many of the HP-IL interface functions can be implemented in a single LSI integrated circuit with the remaining aspects of the protocol implemented in firmware using a standard microprocessor. The microprocessor may also be used to control the device functions and the interaction between the device and the interface. Since most devices will require the microprocessor anyway, its cost is only partly attributable to the interface and partly to the device itself.

1.4 Mechanical Overview

Since HP-IL utilizes a loop structure, each device only needs two connectors, an input and an output, regardless of how many devices are on the loop. In the interest of small size, HP-IL uses a special connector. Both connectors panel mounted together only require about 0.4 inches by 1.5 inches of panel space and about 0.8 inches of depth. If less space than this is available the lines themselves can go directly into the panel through strain reliefs with cable connectors at the loose ends. connectors are constructed so that the reference line cannot be reversed and also so that an output cannot be plugged into another output. Cable connectors plug together properly to form running cable "splices". See figure 1-4.

For relatively short distances up to a maximum of 10 meters from one device to the next, very low cost "zip cord" can be used as shown in the illustration. Note that the "rib" on the "zip cord" always corresponds to the reference line. For longer distances up to 100 meters between devices, shielded twisted pair cable must be used. No change is necessary in the device electronics for the longer distance. Only the cable between those particular devices that are farther than 10 meters apart must be changed.
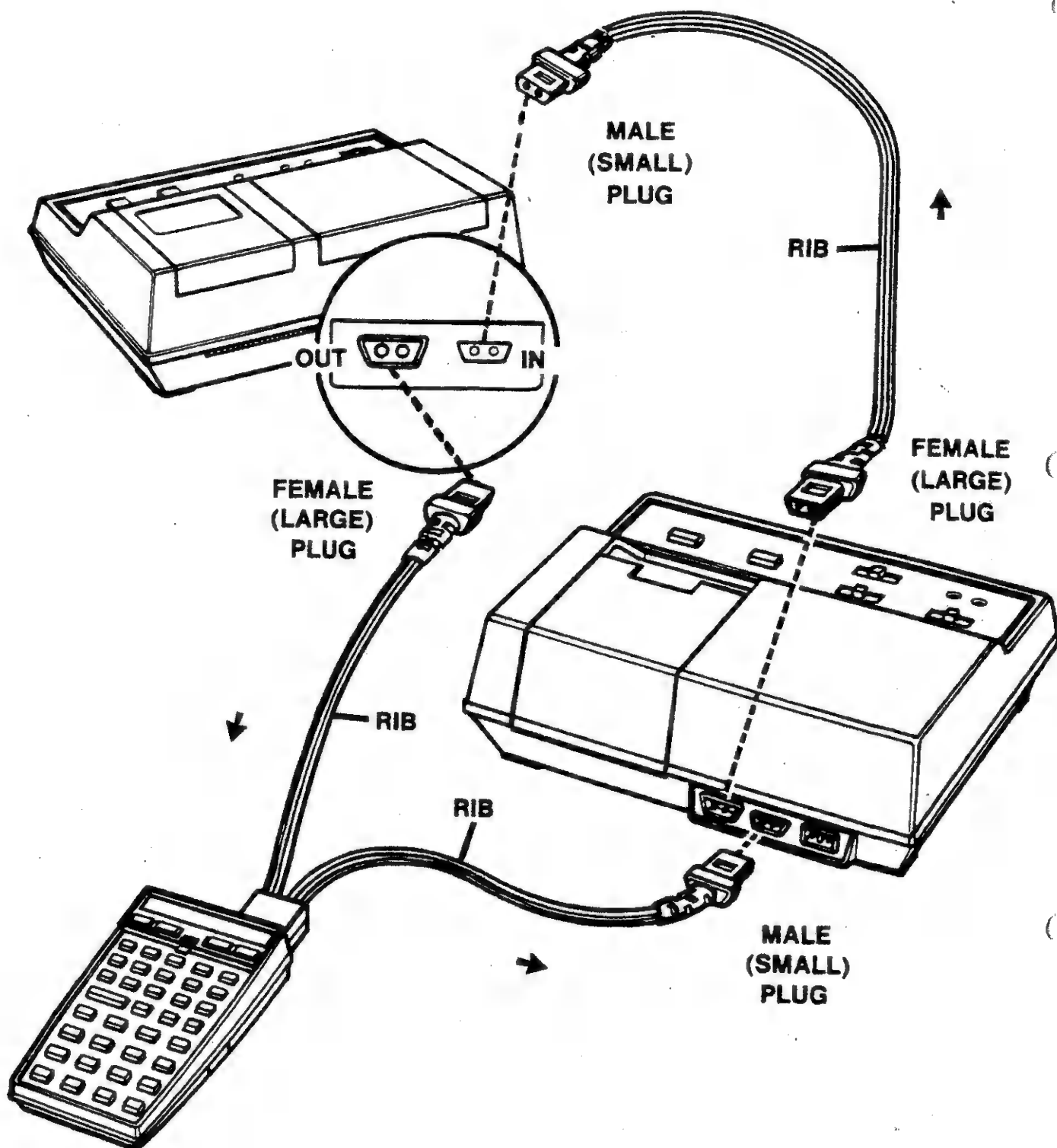
Figure 1-4.   HP-IL Connectors

2. FUNCTIONAL SPECIFICATIONS

2.1 Introduction

The complete functional specification of the HP-IL system is contained throughout this chapter in the state diagrams, associated text, tables, and examples. In cases of apparent disagreement, the state diagrams are to be used as the primary reference with all other material of secondary importance. All cases not specifically accounted for in the diagrams or text are to be ignored.

An HP-IL device may be partitioned conceptually into three functional areas, each with unique capabilities: device functions, interface functions, and message coding logic. Figure 2-1 shows how these functional areas interact within each HP-IL device.

2.1.1 Logical Partitions

Device functions are those unique features and capabilities that each device possesses. They represent the entire motivation for building any device and are specified by the device designer. The device functions can affect the operation of interface functions with local messages.

Interface functions are the basic operational units of the HP-IL system. Through the interface functions, the device may receive, process, and send messages. Interface functions are not specified by the designer, but are defined in this chapter. All interface functions within a physical device must conform to the specifications exactly in order for the device to properly communicate with other devices in the HP-IL system.

The message coding logic performs transformations between logical interface messages and signal line values.
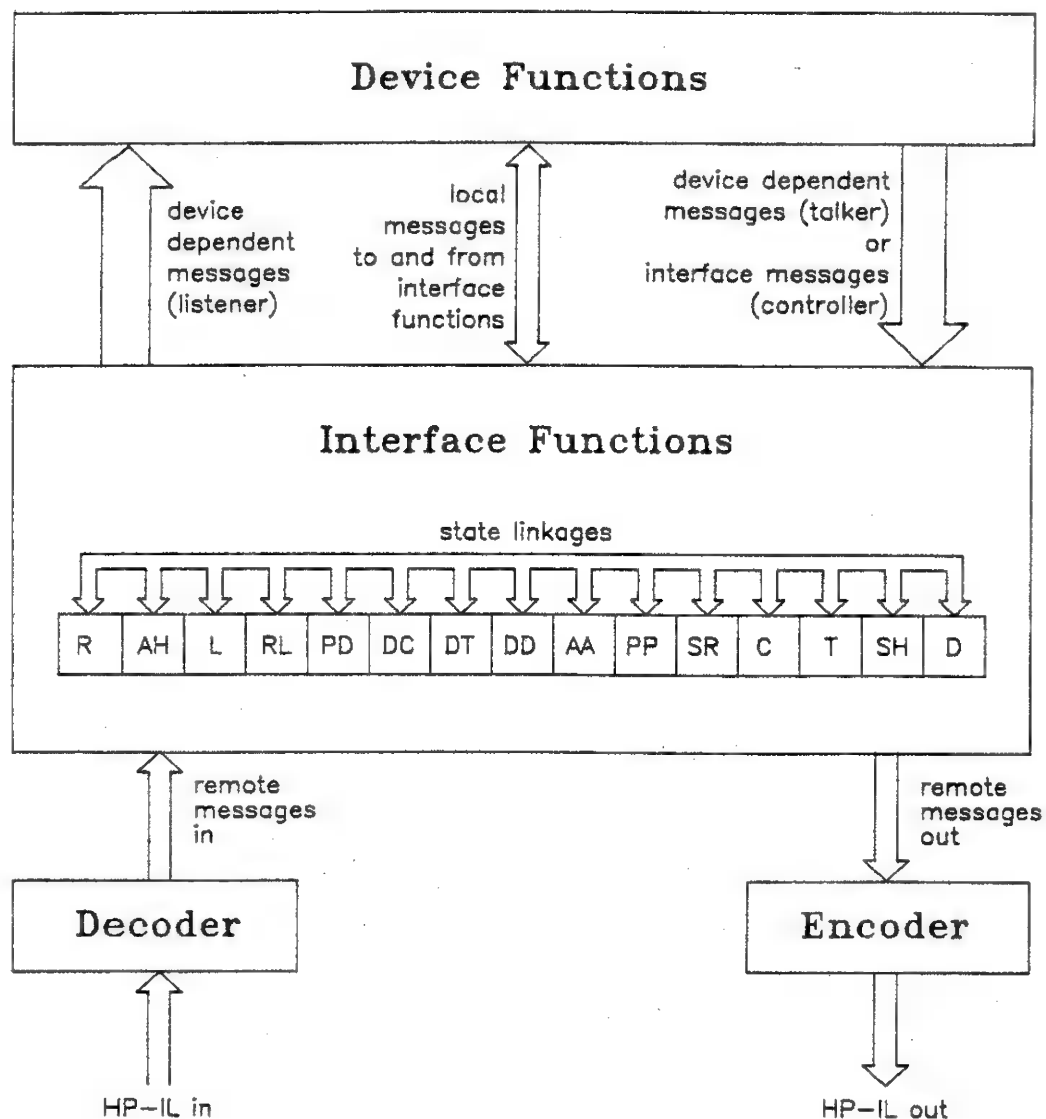
Section		Interface Function	Section		Interface Function
2.3	R	receiver	2.11	RL	remote local
2.4	D	driver	2.12	AA	automatic address
2.5	AH	acceptor handshake	2.13	PD	power down
2.6	SH	source handshake	2.14	PP	parallel poll
2.7	C	controller	2.15	DC	device clear
2.8	T	talker	2.16	DT	device trigger
2.9	L	listener	2.17	DD	device dependent
2.10	SR	service request			command

Figure 2-1.   HP-IL Functional Partitions

The interface functions are defined in terms of one or more groups of interconnected states called state diagrams. The state diagrams define only the external behavior of the interface functions and are not intended to limit the design. Interface functions may be implemented with any combination of hardware, firmware, or software as long as the external behavior is the same. Refer to figure 2-1 for a complete list of interface functions and their mnemonics.

Within each connected group of states, one and only one state may be active at any given time. The state variable associated with a state is true if and only if that state is active. Definitions given for each state describe the messages that must be sent while that state is active and the conditions which cause a transition to another state within the group.

All communications between an interface function and its environment is through messages or combinations of messages. A complete understanding of messages and of figure 2-1 is essential to understanding the state diagrams and the entire interface system. Each message has a logical value associated with it which is either true or false at any specific time. The inputs to an interface function are remote messages, pseudomessages, state linkages, and local messages. The outputs are remote messages and local messages (state variables).

Local messages are those messages sent between the device functions and the interface functions. The set of local messages sent to the interface functions is defined and may not be changed by the device designer. Local messages sent from interface functions to device functions may include any state variable or combination of state variables and are completely designer specified. Local messages sent by device functions must exist for enough time to cause the required state transitions.

Remote messages are translated to or from the interface signal line values through the process of encoding and decoding. Interface functions must ignore any message coding not specifically defined.

Pseudomessages (not shown in figure 2-1) are a small set of messages similar to local messages which are sent by the message coding logic to the interface functions.

A state linkage is a logical interconnection between an active state of one state diagram and a transition between states in another. This is the vehicle for the extensive interaction between the interface functions.

)   Prior to decoding, remote messages consist of eleven-bit
message frames. The three control bits place the message in one
of four major classes, and the eight data bits specify the
particular message within the category. These bits are sent in
the following order and designated as shown:

C2 C1 C0 D7 D6 D5 D4 D3 D2 D1 D0

Note that C2 is also the sync bit indicating the start of a
frame. The major classes of messages are:

1. Data or End messages (DOE). These are device
dependent messages which are sourced by the active
talker and received by the active listener(s). End
messages (control bit C1 set) are data messages that
indicate an end-of-record condition (not end of
transmission). The service request message (control
bit C0 set) indicates to the controller that one or
more devices on the loop need attention.

2. Command messages (CMD). These interface messages
are always sourced by the active controller. Several
groups of commands are defined by the coding of the
data bits. These groups are UCG (universal command
group) messages, ACG (addressed command group)
messages, TAG (talk address group) messages, LAG
(listen address group) messages, and SAG (secondary
address group) messages. All devices respond to UCG,
TAG, and LAG commands, but only devices enabled by a
talk or listen address respond to ACG or SAG commands.

3. Ready messages (RDY). These interface messages are
sometimes sourced by the active controller and
sometimes by the active talker or active listener.
They are used for handshake purposes such as initiating
or terminating a data transmission. Groups of ready
messages are RFC (ready for command group) messages,
ARG (addressed ready group) messages which include SOT
(start of transmission) and EOT (end of transmission)
messages, and AAG (auto address group) messages.

4. Identify messages (IDY). These messages are
normally sent by the controller to determine if devices
have a need for service. Note that bit C0 of both DOE
and IDY messages may be set by devices to indicate a
need for service.

2-4

Detailed definitions of all local and remote messages are given in appendix B. Information regarding coding of remote messages can be found in section 2-18 as well as appendix C.

2.2 State Diagram Notation

Figure 2-2 shows the notation used throughout this chapter within the state diagrams. Each state that an interface function can assume is represented by a four letter upper-case mnemonic always ending in S within a circle. All permissible transitions between states are represented by arrows qualified by expressions that evaluate either true or false. The interface function shall remain in its current state if all expressions which qualify transitions leading to other states are false. The interface function shall enter the state pointed to if and only if the associated expression becomes true.

Notation	Meaning	Notation	Meaning
(TACS)	interface function state	expression →	transition between states
ACDS	linkage from other interface functions	+	OR logical operator
PPU	remote messages from the interface	•	AND logical operator
rdy	local messages from the device	$\overline{\text{SPAS}}$	NOT logical operator
sync	pseudomessages from the encoding/ decoding logic	[+ SDC • LACS]	optional expression term (at the designer's choice)

Figure 2-2.   State Diagram Notation

2-5

An expression consists of one or more local messages, pseudomessages, remote messages, or state linkages used in conjunction with the operators AND, OR, or NOT. Local messages to an interface function are represented by lower-case three-letter mnemonics. Pseudomessages are represented by lower-case four-letter mnemonics. Remote messages are represented by upper-case three-letter mnemonics. A linkage from another state diagram is represented by the state mnemonic in boldface type. A state linkage is true if the state is currently active; otherwise, it is false. The AND operator is represented by a dot. The OR operator is represented by a plus sign. The NOT operator is represented by a horizontal bar placed over the portion of the expression to be negated. The resulting negated expression has a true value if and only if the value of the expression under the bar is false. The AND operator takes precedence over the OR operator in an expression unless indicated otherwise by parentheses.

If a portion of an expression is optional in that its true value is not required for the complete expression to be true (at the designer's choice), then it is enclosed within square brackets. If a specific expression causes a transition to a state from all other possible states of the diagram, an arrow without a state at its origin is used, and is assumed to originate in all states. Note also that the power-off state is a valid state of most interface functions and should normally be shown with a transition leading to the state to be entered when power is first turned on. This state is omitted for clarity in the diagram and only the transition to the first state is shown.

2.3 R (Receiver) Function

The R interface function provides the device with the capability to receive messages over the interface. It categorizes the received message into major classifications which are used by the AH (Acceptor Handshake) and D (Driver) interface functions to effect proper communication.

The R function receives the remote messages one bit at a time and decides as quickly as possible if immediate retransmission is required. If so, the message is passed directly to the D interface function for retransmission. The R function also decides whether the message is to be interpreted by the device or other interface functions or is to be ignored. If the device is to interpret the message, it is passed to the AH function. If no device or interface interpretation is required, the frame is passed to the D function. When the D or AH function signals that it has accepted the message, the R function returns to its idle state ready to receive the next remote message.

While it is possible to simplify the implementation of the R function by waiting for all message bits to be available before performing any decoding, the resulting throughput would be unacceptable. It is strongly recommended that the decoding begin as soon as enough bits have been received to permit correct decoding. This occurs on the first bit for DOE messages, the second bit for IDY, the third bit for CMD, etc. In general, only a few bits need be received to decide if retransmission is necessary.

Please note that all devices must implement the full R interface function capability. No subsets are permitted.

PONS

REIS

sync

$\overline{\text{CMD}} \cdot \text{DTRS}$ ACDS + NRD $\cdot$ NRWS $\cdot$ ACRS

RSYS

echo $\cdot \overline{\text{DTRS}}$ hold

RITS   CMD $\cdot$ DTRS   RCDS
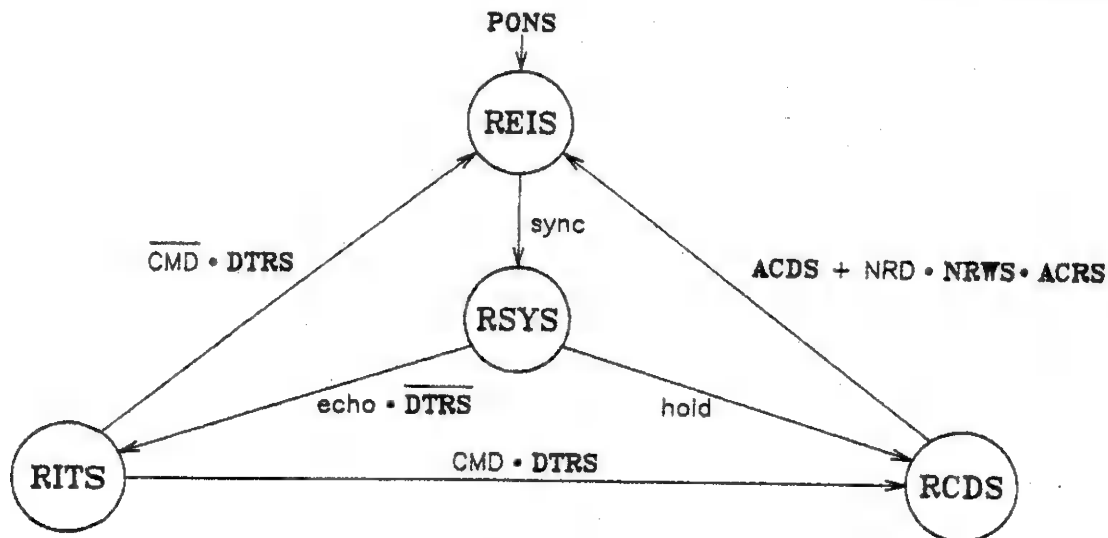
echo = DOE $\cdot \overline{\text{TACS}} \cdot \overline{\text{SPAS}} \cdot \overline{\text{DIAS}} \cdot \overline{\text{AIAS}} \cdot \overline{\text{TAHS}} \cdot \overline{\text{TERS}} \cdot \overline{\text{LACS}} \cdot \overline{\text{NRWS}} \cdot \overline{\text{CACS}}$

   + (CMD+IDY) $\cdot \overline{\text{CACS}} \cdot \text{CSBS}$

   + ARG $\cdot \overline{\text{TADS}} \cdot \overline{\text{TACS}} \cdot \overline{\text{SPAS}} \cdot \overline{\text{DIAS}} \cdot \overline{\text{AIAS}} \cdot \overline{\text{TAHS}} \cdot \overline{\text{TERS}} \cdot \overline{\text{TPAS}} \cdot \overline{\text{LACS}} \cdot \overline{\text{CACS}} \cdot \overline{\text{CSBS}}$

   + (AAD+AMP) $\cdot \overline{\text{AAUS}}$ + AEP $\cdot \overline{\text{AWPS}}$ + AES $\cdot \overline{\text{ZES}} \cdot \text{AWSS}$ + AES $\cdot \overline{\text{AAUS}} \cdot \overline{\text{AWSS}}$

   + IAA + IEP + IMP + IES

hold = DOE $\cdot$ (TACS+SPAS+DIAS+AIAS+TAHS+TERS+LACS+NRWS+CACS)

   + RDY $\cdot \overline{\text{ARG}} \cdot \overline{\text{AAG}}$ + (CMD+IDY) $\cdot$ (CACS+CSBS)

   $\cdot$ + ARG $\cdot$ (TADS+TACS+SPAS+DIAS+AIAS+TAHS+TERS+TPAS+LACS+CACS+CSBS)

   + (AAD+AES+AMP) $\cdot$ AAUS + AEP $\cdot$ AWPS + ZES $\cdot$ AWSS

Messages

sync   valid sync bit received

AAD	auto address
AAG	auto address group
AEP	auto extended primary
AES	auto extended secondary
AMP	auto multiple primary
ARG	addressed ready group
CMD	command

DOE	data or end
IAA	illegal auto address
IDY	identify
IEP	illegal extended primary
IES	illegal extended secondary
IMP	illegal multiple primary
NRD	not ready for data
RDY	ready
ZES	zero extended secondary

Interface States

RCDS	receiver data state (links to AH,C,L)
REIS	receiver idle state
RITS	receiver immediate transfer state (links to D,C,SR,PP)
RSYS	receiver sync state
AAUS	auto address unconfigured state (from AA)
ACDS	acceptor data state (from AH)
ACRS	acceptor ready state (from AH)
AIAS	accessory ID active state (from T)
AWPS	auto wait for primary state (from AA)
AWSS	auto wait for secondary state (from AA)
CACS	controller active state (from C)

CSBS	controller standby state (from C)
DIAS	device ID active state (from T)
DTRS	driver transfer state (from D)
LACS	listener active state (from L)
NRWS	not ready wait state (from L)
PONS	power on state (from PD)
SPAS	serial poll active state (from T)
TACS	talker active state (from T)
TADS	talker addressed state (from T)
TAHS	talker hold state (from T)
TERS	talker error state (from T)
TPAS	talker primary active state (from T)

Figure 2-3. R State Diagram

REIS (Receiver Idle State)

        In REIS the R interface function is not engaged in any
message transfer process. The function powers on in this state.
Transition out of this state does not occur until a valid sync
bit has been received from the loop.

RSYS (Receiver Sync State)

        In this state the R interface function has begun to
receive a message from the interface and is in the process of
determining whether to immediately retransmit the message (echo)
or not (hold).

RITS (Receiver Immediate Transfer State)

        DOE messages not sourced by or intended for this device,
CMD or IDY messages not sourced by this device, and ARG messages
not for this device should be immediately retransmitted on the
loop. In RITS the R function is signaling the D (Driver)
function that retransmission of the incoming frame should
commence. RITS is not entered until the D function is finished
transmitting any previous message and is not exited until the D
function has accepted the current message and commenced
retransmission. Most messages which are echoed do not affect the
device or interface functions (except R and D), however CMD
messages are both echoed and held for device and interface
interpretation. This decoding is also done in RITS. If the
message is not a CMD the function returns to REIS, otherwise the
transition is to RCDS.

RCDS (Receiver Data State)

        In this state the R function is signaling the AH
(Acceptor Handshake) function that the message being received is
for this device. When the AH function signals in return that it
has accepted the message, the R function returns to REIS.

2.4   D (Driver) Function

The D interface function provides the device with the
ability to transmit messages on the loop. These messages come
from three different sources: messages generated by this device,
messages received on the loop which must be immediately
retransmitted, and messages received on the loop which must be
retransmitted after acceptance by the device.

When the R (Receiver), SH (Source Handshake), or AH
(Acceptor Handshake) functions indicate that a message is
available for transmission, the D function begins the
transmission. When the frame transmission is complete, the D
function again becomes idle.

The D function transmits all remote messages. These
messages are transferred to it by the R, AH, or SH functions. In
this transfer certain modifications of the message may be
performed by the parallel poll and service request interface
functions. The SR function allows the device to set the SRQ bit
on DOE or IDY frames before the Driver transmits them on the
loop. The PP function allows the device to modify one bit of IDY
frames before retransmission for parallel poll response. Refer
to these two functions for a full description of this
modification process.

All devices must have the full D function implemented.
No subsets are permitted.
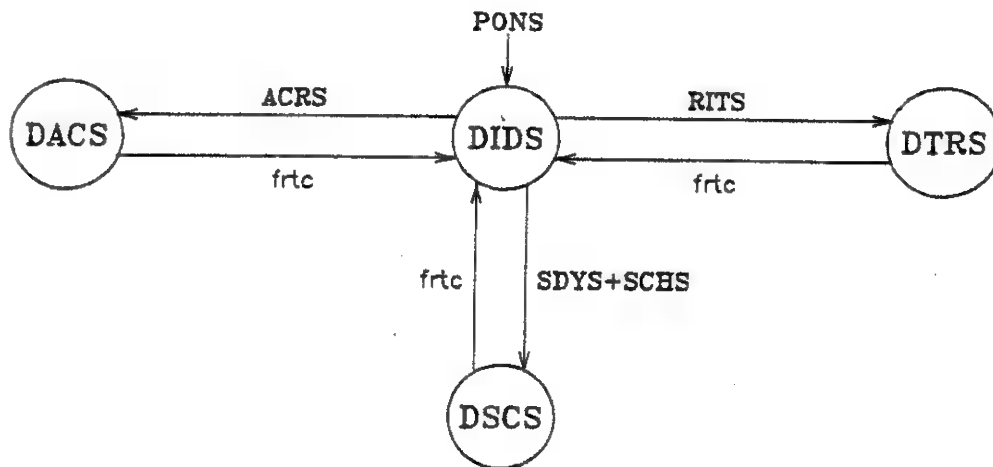
Messages

frtc   frame transmission complete

Interface States

DACS   driver transmit from acceptor state
(links to AH,L)

DIDS   driver idle state (links to SR,PD)

DSCS   driver transmit from source state
(links to SH)

DTRS   driver transfer state (links to R,PP)

ACRS   acceptor ready state (from AH)

PONS   power on state (from PD)

RITS   receiver immediate transfer state
(from R)

SCHS   source command handshake state
(from SH)

SDYS   source delay state (from SH)

Figure 2-4.   D State Diagram

2-11

DIDS (Driver Idle State)

In DIDS the D function is not transmitting any message. The function powers on in this state. When the SH (source handshake), AH (acceptor handshake), or R (receiver) functions indicate that a frame is ready for transmission, the D function enters DSCS, DACS, or DTRS respectively.

DSCS (Driver Source State)

In this state the function is transmitting a message frame originated by this device. DSCS is entered when the SH (Source Handshake) function indicates that the message is ready to be transmitted. When transmission is complete the function returns to DIDS.

DTRS (Driver Transfer State)

In DTRS the D function has commenced retransmitting a message transferred from the R (Receiver) function which must not be held up by the device's ability to interpret it. When finished with the retransmission the D function returns to DIDS.

DACS (Driver Acceptor State)

This state is entered when retransmission of a message which has been received and interpreted by the device has begun. The message is transferred from the AH (Acceptor Handshake) function. When transmission is complete the function transitions to DIDS.

2.5 AH (Acceptor Handshake) Function

The AH interface function provides the device with the capability to interpret remote messages. It also determines if the frame needs to be retransmitted after interpretation (repeat) or not (norepeat), and indicates to the D (Driver) interface function when this retransmission should begin.

If the Receiver interface function signals that a frame has been received and is intended for this device, the Acceptor Handshake function indicates to the device and to the other interface functions when that frame is ready for interpretation. If the message is to be retransmitted, the transmission begins after the device indicates that it has finished interpreting the message (with the local message rdy). The AH function also controls the NRD (not ready for data) sequence by holding the data while the device sources the NRD message and retransmitting the data when the NRD returns.

The AH interface function is not capable of originating remote messages. It only accepts received messages from the R (Receiver) interface function and transfers them to the device and other interface functions for appropriate action. After the device indicates that interpretation is complete, the AH function can transfer the received message to the Driver interface function for retransmission.

The AH function may be implemented such that it receives all eleven bits of the message frame prior to any decoding (repeat, norepeat expressions). It is permitted, however, to perform this decoding earlier in the frame so that loop throughput may be considerably enhanced at the expense of greater complexity.

Please note that all devices must implement the full AH capability. No subsets are permitted.

PONS
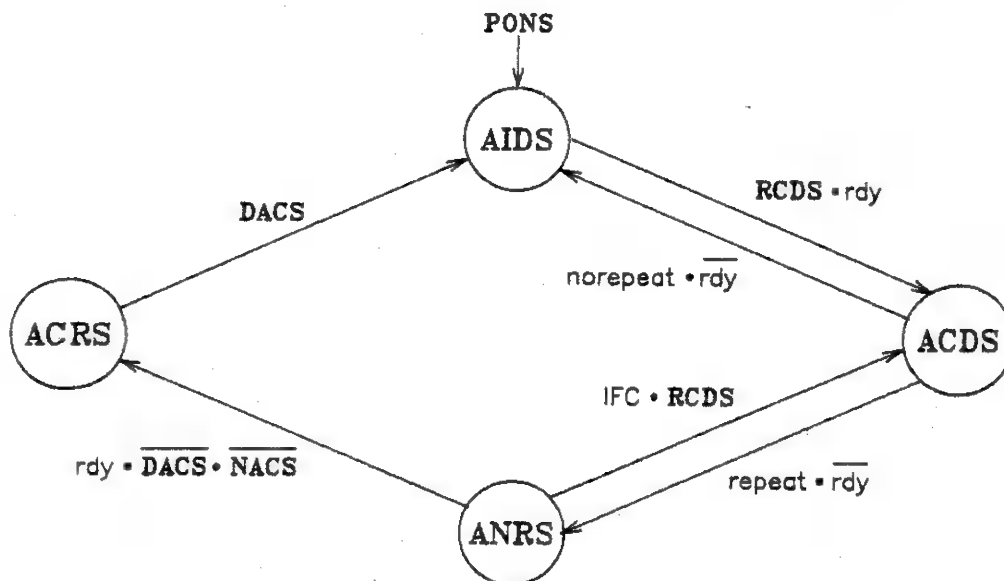
$$repeat = DOE \cdot (\underline{LACS} + \underline{NRWS} + \underline{CACS})$$
$$+ RDY \cdot \overline{SOT} \cdot \overline{AAG} \cdot \overline{CACS} \cdot \overline{CSBS}$$
$$+ SOT \cdot \overline{TADS} \cdot \overline{TACS} \cdot \overline{SPAS} \cdot \overline{DIAS} \cdot \overline{AIAS} \cdot \overline{TAHS} \cdot \overline{TERS}$$
$$+ NRD \cdot \overline{CACS}$$

$$norepeat = DOE \cdot (TACS + SPAS + DIAS + AIAS + TAHS + TERS)$$
$$+ CMD + IDY + AAG$$
$$+ RDY \cdot \overline{NRD} \cdot (CACS + CSBS)$$
$$+ SOT \cdot (TADS + TACS + SPAS + DIAS + AIAS + TAHS + TERS)$$
$$+ NRD \cdot CACS$$

Messages

rdy	ready for next frame	IDY	identify
		IFC	interface clear
AAG	auto address group	NRD	not ready for data
CMD	command	RDY	ready
DOE	data or end	SOT	start of transmission

Interface States

ACDS	acceptor data state (links to R,SH,C,T,L, SR,RL,AA,PD,PP,DC,DT,DD)	DIAS	device ID active state (from T)
		LACS	listener active state (from L)
ACRS	acceptor ready state (links to R,D)	NACS	not ready active state (from L)
AIDS	acceptor idle state (links to PD)	PONS	power on state (from PD)
ANRS	acceptor not ready state	RCDS	receiver data state (from R)
		SPAS	serial poll active state (from T)
AIAS	accessory ID active state (from T)	TACS	talker active state (from T)
CACS	controller active state (from C)	TADS	talker addressed state (from T)
CSBS	controller standby state (from C)	TAHS	talker hold state (from T)
DACS	driver transmit from acceptor state (from D)	TERS	talker error state (from T)

Figure 2-5.   AH State Diagram

AIDS (Acceptor Idle State)

In AIDS the AH function is not engaged in the handshake cycle and does not have a message available. This is the power-on state for this function. When the R (Receiver) function indicates that a message for this device is available, the AH function enters ACDS.

ACDS (Acceptor Data State)

In this state the interface function is indicating to all other interface and device functions that a message for this device has been received and is now valid for interpretation. This state also decodes the message to determine whether it should be retransmitted or not. DOE, CMD, RDY or IDY messages sourced by this device, and SOT messages intended for this device are not retransmitted on the loop. After the device has indicated that it has accepted the message for interpretation by sending the rdy local message false (not ready), the AH function returns to AIDS if retransmission is not necessary, or transitions to ANRS if the message must be retransmitted.

ANRS (Acceptor Not Ready State)

In ANRS the AH function is waiting for the device to indicate (via rdy) that it is ready for the next message frame. This typically occurs after the device has completed any actions necessitated by the current message. When the device is ready the function transitions to ACRS. If the R function indicates that the IFC message has been received while the AH function is waiting in ANRS, the function transitions back to ACDS.

ACRS (Acceptor Ready State)

This state is indicating to the D (Driver) function that retransmission of the current message frame should begin. When the D function indicates in return that it has accepted the frame and has commenced transmission, the AH function returns to AIDS.

2.6 SH (Source Handshake) Function

         The SH interface function gives the device the capability
to originate remote messages primarily while the device is a
talker or controller. It coordinates the message transfer from
the device to the interface.

         For normal loop handshake, the SH interface function
accepts a message from the device and transfers it to the D
interface function. When the R function indicates that the
transmitted frame has returned, the SH function completes the
handshake with the device and becomes ready to accept another
message. For sourcing commands, the SH function causes the RFC
(ready for command) message to be sent after every command is
sent and received. The SH function also controls asynchronous
handshaking with the local message lab (local abort).

         The SH function becomes ready to accept messages from the
device when the device is first enabled to source remote messages
(source expression). If the SH function is ready and the device
has a message available, message transmission begins and the
function waits for the message to return. When the R function
indicates that the frame has returned, the SH function again
becomes ready to source another frame. As soon as the device is
no longer enabled to source messages, the source handshake
function returns to idle.

         The SH function does not originate remote messages on the
loop but merely controls the orderly transfer of messages from
the device to the D (Driver) function for transmission. The
device may change the message frame while SIDS or SGNS are
active, but must hold the frame valid while in SDYS and STRS.

         All devices must have the full SH function. No subsets
are permitted.
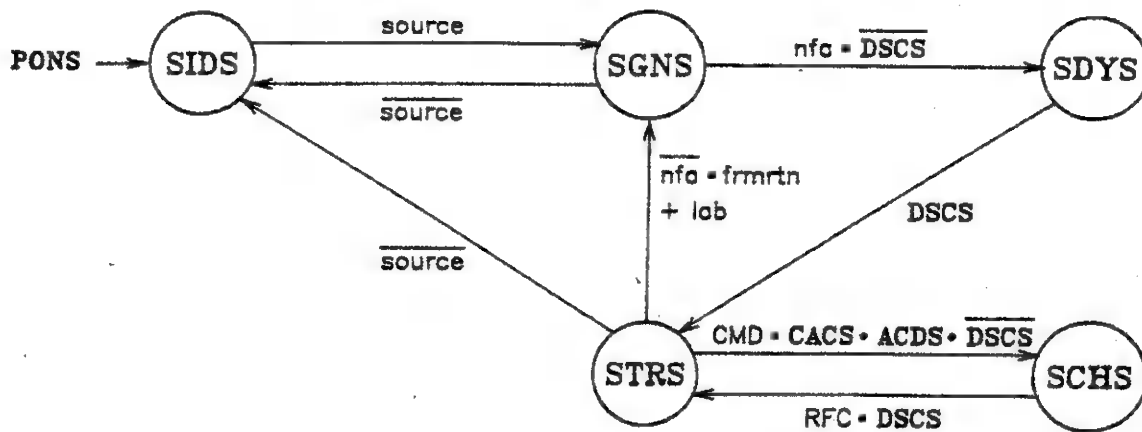
$$source = TACS + SPAS + DIAS + AIAS + TAHS + TERS + CACS$$
$$+ NACS + ARSS + AAIS + ASIS + APIS + AMIS$$

$$frmrtn = DOE \cdot (TACS + SPAS + DIAS + AIAS + TAHS + TERS) \cdot ACDS$$
$$+ (RDY+IDY) \cdot CACS \cdot ACDS$$

Messages

lab	local abort		DOE	data or end
nfa	new frame available		IDY	identify
			RDY	ready
CMD	command		RFC	ready for command

Interface States

SCHS	source command handshake state (links to D)		ASIS	auto secondary increment state (from AA)
SDYS	source delay state (links to D)		CACS	controller active state (from C)
SGNS	source generate state		DIAS	device ID active state (from T)
SIDS	source idle state		DSCS	driver transmit from source state (from D)
STRS	source transfer state (links to C,T,SR,AA)		NACS	not ready active state (from L)
AAIS	auto address increment state (from AA)		PONS	power on state (from PD)
AIAS	accessory ID active state (from T)		SPAS	serial poll active state (from T)
ACDS	acceptor data state (from AH)		TACS	talker active state (from T)
AMIS	auto multiple increment state (from AA)		TAHS	talker hold state (from T)
APIS	auto primary increment state (from AA)		TERS	talker error state (from T)
ARSS	asynchronous request source state (from SR)			

Figure 2-6.   SH State Diagram

SIDS (Source Idle State)

        The SH function powers on in SIDS and in this state no
handshake is taking place. The device is not sourcing frames.
When the appropriate interface functions indicate that the device
should begin to source messages (source expression) the function
transitions to SGNS.

SGNS (Source Generate State)

        In SGNS the device is in the process of generating the
next message frame to be sourced. When the device indicates that
the new message is available (nfa local message) and the D
(Driver) function indicates that it is not still transmitting a
previously sourced frame, the function enters SDYS. When the
sourcing device is no longer enabled to source messages, the
function will return to SIDS.

SDYS (Source Delay State)

        In this state the SH function is waiting for the D
(Driver) interface function to acknowledge receipt of the sourced
message and begin transmission. When this occurs the function
enters STRS.

STRS (Source Transfer State)

        In STRS the SH function is waiting for the sourced
message to go around the loop and return (normal loop handshake,
frmrtn expression). When this occurs the function returns to
SGNS if the sourced message was not a command and transitions to
SCHS if the message was a command. Asynchronous loop operations
are possible using the lab local message to force the return to
SGNS before the handshake is complete so a new message frame may
be sent. Normally, the last frame sourced by a device will be a
Ready frame (such as EOT if talker) and will cause a transition
directly from STRS to SIDS since the frame might not return to
the device.

SCHS (Source Command Handshake State)

        The function enters SCHS when the receiver function
indicates that the sourced command has returned. In this state,
the function is signaling to the driver function that the RFC
(ready for command) message is ready for transmission. When the
driver function indicates that the RFC transmission has begun, the
function returns to STRS to wait for the RFC to return.

2.7 C (Controller) Function

The C function provides a device with the ability to send
interface messages (including CMD, RDY, and IDY messages) to
devices on the loop. It also provides the capability for
conducting parallel polls, detecting the SRQ message from devices
requesting service, and detecting transmission errors in device
dependent data transmissions. These capabilities exist only
while the controller function is in its active state.

If more than one device that possesses the C function is
on the loop, then all but one must be in the idle state at any
given time. The device with the active C function is called the
active controller. Protocol is provided to allow controllers to
take turns as active controller of the loop.

One and only one controller device may send the local
message scl true indicating that it is the system controller.
This condition should remain true throughout the operation of the
interface. The system controller takes charge at power on and is
the only device which can source the IFC message. It can do this
at any time regardless whether it is the active controller.

When this function is in its active state, the device is
enabled to originate interface messages including CMD, RDY, and
IDY message frames under control of the SH (Source Handshake)
function. These interface messages may be used to control the
interface functions within other devices on the loop. When the C
function is not in its active state, the device may not source
interface messages. The C function does not give the device the
capability to source or accept DOE (data or end) messages. Those
capabilities are offered only by the T and L functions.

The normal loop handshake consists of sourcing one frame
at a time and waiting for its return while each receiving device
holds the frame until it is ready for the next. When a frame
returns, all devices on the loop are ready to receive the next
message. For frames intended for only a few devices, the penalty
of such serial interpretation is not severe. However, since CMD
frames are often intended for all devices, they are retransmitted
immediately by each receiving device to allow for parallel
interpretation. When a CMD frame returns, the active controller
sources the RFC ready frame which is held by each device until it
is ready for the next message. Refer to the SH function for more
information about both types of loop handshake.

2-19

It is strongly recommended that the controller error check its messages by comparing the received frame with the previously transmitted frame to guarantee proper reception of messages and to enhance reliability.

The active controller may source the IDY message asynchronously to the loop handshake at any time but will not normally do so unless there has been a long delay in loop transmission. All devices should be implemented so that they will respond properly to the IDY and yet still continue correct response to other messages without disturbing normal loop operation. The controller may use the local message gta to enable the asynchronous IDY if it is not already in the active state. Refer to chapter 5 for complete discussion of asynchronous operations.

Devices on the loop indicate a need for service from the controller by setting the SRQ bit in DOE or IDY frames. When rapid response is important, the controller may execute a parallel poll by sending the IDY message. When configured to do so, devices respond by modifying one of the assigned eight data bits in the IDY frame. Devices may also have the capability of asynchronously sourcing their own IDY frames with the SRQ bit set. The controller may choose to enable this mode when no loop operations are in progress.

For various reasons the controller may choose to interrupt a device dependent message transmission between a talker and listener(s). This may be done through the L (listener) interface function. The controller may also enable active listeners to perform this NRD sequence so that the transfer may be halted before extensive delays occur. Refer to section 2.9 for further information about this capability.

The system controller is the only device on the loop which is permitted to source the IFC message. It may do this at any time (even if it is not the currently active controller) and take control of the loop asynchronously. Under this circumstance it is possible to have an extraneous frame on the loop since the source of the frame may be cleared before the frame returns. To prevent this frame from recirculating, the system controller, after sourcing the IFC, must destroy any frames which it receives until the IFC returns. Further, no device may source any frame after retransmitting the IFC (and RFC) until specifically commanded to do so by succeeding interface messages. The sic or gta local messages may initiate this sequence depending on the state of the system controller.

When power is first applied the system controller should observe special protocol to insure that all devices on the loop are present and properly connected. The recommended sequence is to send the IFC message at regular intervals while stopping all frames received to guarantee that no extra frames are circulating. When one IFC returns, loop continuity is verified. At that time, the controller should send one and only one RFC while stopping all frames being received. When the RFC returns, the loop is ready for normal operations. Refer to chapter 5 for more information on the power up sequence.

No controller capability is designated C0. If the device has the controller capability, it must include subset C1. The allowable subsets of the controller function are:

C1:   Basic controller capability. Includes the ability to send interface messages and detect transmission errors by interpreting the ETO or ETE messages. Requires CIDS, CACS, CSBS, CEIS, and CEMS.

C2:   System controller capability. Provides the capability to send the Interface Clear message. If this capability is not present the local messages sic and scl must always be false. It is strongly recommended that the scl message not always be true so that the device may be used on a loop with multiple controllers.

C3:   Detect SRQ capability. Note that the L function is necessary if the device is to receive and interpret status bytes. Requires CSNS and CSRS.

C4:   Pass and receive control capability. This allows controllers to take turns controlling the loop. Requires CTRS and the optional term in the transition between CIDS and CACS. The T function is also required.

C5:   Parallel poll capability. Includes the ability to conduct parallel polls and to enable device response.

C6:   Asynchronous service request capability. Includes the ability to enable devices to source asynchronous IDY messages and to interpret those messages.

All controllers will include C1 with, optionally, some combination of the other capabilities. A simple system controller with only SRQ capability would be C1,2,3. A basic controller with parallel poll and control passing capability would be designated C1,4,5.
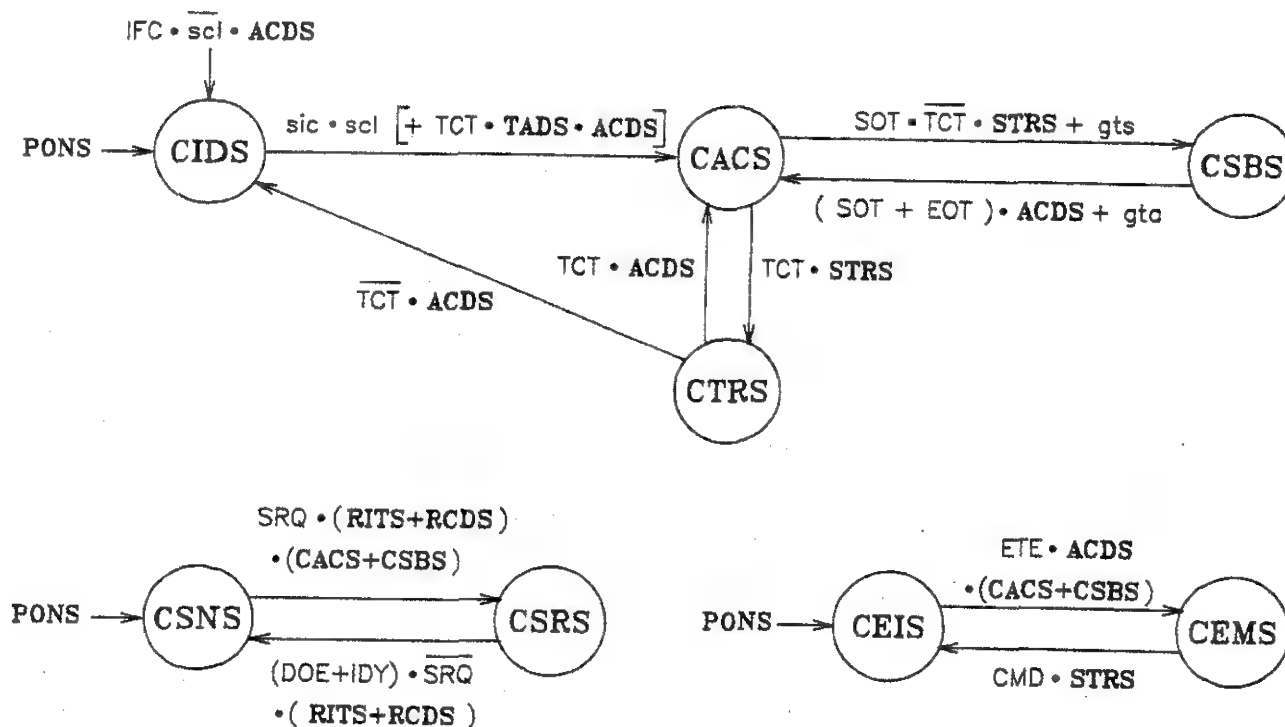
Messages

gta	go to active	EOT	end of transmission
gts	go to standby	ETE	end of transmission with error
scl	system control	IFC	interface clear
sic	send interface clear	SOT	start of transmission
		SRQ	service request
CMD	command	TCT	take control

Interface States

CACS	controller active state (links to R,AH, SH,T,L,PD)	ACDS	acceptor data state (from AH)
CEIS	controller error idle state	PONS	power on state (from PD)
CEMS	controller error mode state	RCDS	receiver data state (from R)
CIDS	controller idle state	RITS	receiver immediate transfer state (from R)
CSBS	controller standby state (links to R,AH,L)	STRS	source transfer state (from SH)
CSNS	controller service not requested state	TADS	talker addressed state (from T)
CSRS	controller service requested state		
CTRS	controller transfer state		

Figure 2-7.   C State Diagram

CIDS (Controller Idle State)

In CIDS the device has no loop control capability. This is the power-on state. If the function receives the TCT message while addressed to talk or the device is system controller and has need of sending the IFC (interface clear) message (local messages scl and sic), then the function enters CACS.

CACS (Controller Active State)

In CACS the device is enabled to send interface messages through the SH (source handshake) function. If the function enables a talker to begin sourcing DOE (data or end) messages or the local message gts (go to standby) is true, the function enters CSBS. If the function sends the TCT (take control) message in an attempt to pass active control to another device on the loop, the function enters CTRS.

CSBS (Controller Standby State)

In CSBS the function has enabled a data transfer between a talker and listener(s) with an SOT (start of transmission) message. If the SOT returns or an EOT (end of transmission) is received, the function returns to CACS. While in CSBS the device may not source any remote messages unless it is the active talker. The function may return active with the gta (go to active) local message if asynchronous operations are desired.

CTRS (Controller Transfer State)

In CTRS the device has sent the TCT in an attempt to pass active control to another loop device. If the TCT returns, the attempt is unsuccessful and the device must resume active control of the loop. If any other frame is received, the attempt was successful and this function must enter CIDS.

)

CSNS (Controller Service Not Requested State)

        In this state the function is indicating that no loop
devices are currently requesting service. This is the power-on
state. If a DOE (data or end) or IDY (identify) message is
received with the SRQ bit true, the function enters CSRS.

CSRS (Controller Service Request State)

        In CSRS the function is indicating that at least one loop
device is requesting service. If a DOE or IDY frame is received
with the SRQ bit false, then the function returns to CSNS.

)

CEIS (Controller Error Idle State)

        In this state the function is indicating to the device
that no error has been detected by the talker in a data transfer.
If the ETE message is received the function moves to CEMS.

CEMS (Controller Error Mode State)

        In CEMS the function is indicating to the device that an
talker has discovered an error in a data transfer. When the
controller sends its next command the function returns to CEIS.

)

)

2.8 T (Talker) Function

        The T interface function provides the device with the capability to send device dependent data, status, device ID, and accessory ID on the loop. This capability exists only when the device has been addressed to talk.

        This function enables the device to send device dependent DOE (data or end) messages on the loop under the control of the SH (Source Handshake) interface function. The device may send data messages only if one of TACS, SPAS, DIAS, or AIAS is active.

        To enhance device compatibility it is strongly recommended that wherever possible device dependent data messages be sent using the ASCII code.

        The talker may send the END message within the transmitted data string to indicate an end-of-record condition to the listener(s) without terminating the transmission. The EOT (end of transmission) message should be sent to the controller when the talker has no more data ready to send. The EOT message does not necessarily indicate an end of message condition, it is used only as a handshake to the controller. Therefore, if a delay may occur before the talker is able to source another data byte, ETO should be sent.

        The controller or an active listener (if enabled by the controller) may elect to interrupt the data transfer using an NRD sequence as controlled by the L (Listener) and AH (Acceptor Handshake) interface functions. The talker's current data byte is held by the device interrupting the transfer and the NRD message is inserted in its place. When the talker receives and retransmits the NRD frame, it waits for the held data byte to be returned. When the interrupting device receives its NRD, then the held data byte is retransmitted to the talker. When the talker receives the data byte, it will source the appropriate EOT (end of transmission) message. Refer to the L interface function in section 2.9 for a more complete description.

        Each time a talker becomes enabled to source status or ID information with the SST, SAI, or SDI ready messages, it must begin with the first byte regardless of any previous operations. However, if a talker is enabled to source data with an SDA ready message, the transmission must begin with the current data byte.

A talker must have the capability to be interrupted one or more times during a data transfer without adverse effects. During this interruption, the controller may ask devices on the loop for status or ID information one or more times, or may simply start a transfer between other devices. The talker must be able to be interrupted in this manner without affecting the data to be sourced. When the talker is re-enabled to source data, it must continue at the point of interruption unless directed otherwise by the controller in a device dependent manner.

A device's status response may contain one or more bytes that indicate various device dependent conditions. The two most significant bits of the first byte are defined and may not be used for any other purpose. Bit 7 (msb) is set if the byte is a system status byte and clear otherwise. Bit 6 is always equal to the value of the local message rsv. Refer to the SR (Service Request) interface function on page 2-36 and section 2.18 for information on status message coding.

The device ID is an ASCII string representing the the device's model identification followed by carriage return and linefeed. It is typically of the following form: two alpha characters representing the company code, a one to five digit model number, and a single alpha character model revision. Devices may optionally send additional information (before the carriage return) to indicate such things as options installed in the device or other information to identify or classify the device capabilities.

The accessory ID consists of one or more bytes that provide information about device functions and capabilities. Refer to section 2.18 for a complete description of the accessory ID byte(s).

When a DOE message returns to the talker the SRQ bit may be set. If the talker is not also the controller, the SRQ message must be sent to the active controller. The talker may do this by sourcing the following DOE message (if there is one) with the SRQ bit set. The talker must not allow more than one DOE frame to be sourced without the SRQ bit set after receiving a DOE with the SRQ bit set. If the SRQ bit on any received DOE message is received false, the talker must discontinue sourcing the SRQ message with its device dependent data.

The loop structure is ideal for error checking purposes since the originator of a message can compare the returned frame with the original transmitted frame. While not specifically required it is strongly recommended that devices perform this error checking function to enhance loop reliability. If a device detects an error it should set the local message fre true. The T function will then complete the current handshake and source the ETE (end of transmission with error) message to signal the controller that a transmission error has occured. If the device does not perform error checking, it must always send the fre local message false.

As with other interface functions, a complete understanding of the four frame handling functions (Receiver, Acceptor Handshake, Source Handshake, and Driver) is essential for correct implementation of the Talker function.

A talk-only listen-only system has no controller and involves some special protocol. The talker in such a system must not send the EOT message (since it would circulate endlessly) and therefore must always send the local messages lfs and fre false. Furthermore, after power-on, if the talker has not received back its first message frame after a certain amount of time (say, one second) it should continue to send this message at this slow, device dependent interval until the frame returns completing the loop handshake. Then the talker may begin normal transmission. Without this provision the first frame might be destroyed at an as yet unpowered listener and the talker would never send the next frame. This permits devices in this type of system to be powered on in any order.

If the talker function is not implemented, the device is designated TØ. The designer may choose among several possible subsets of the T function, all of which (except TØ) must include TIDS, TADS, TAHS, TERS. The data, status, accessory ID, and device ID capabilities are designated by the numbers 1, 2, 3, and 4 respectively. It is very strongly recommended that devices respond to both accessory ID and device ID. Talk-only mode (designated T5) requires subset T1 and allows the ton local message to be true. Some means of disabling the ton local message must be provided. Extended addressing may be added with the TPIS and TPAS states (requires the AA2 or AA3 subsets of the auto address function). For example, a device which can send data and status only would be given the T1,2 designation. The same device with the inclusion of talk-only mode would be listed as T1,2,5. A device that can send only status and its accessory ID and which has extended addressing capability would be T2,4,6.
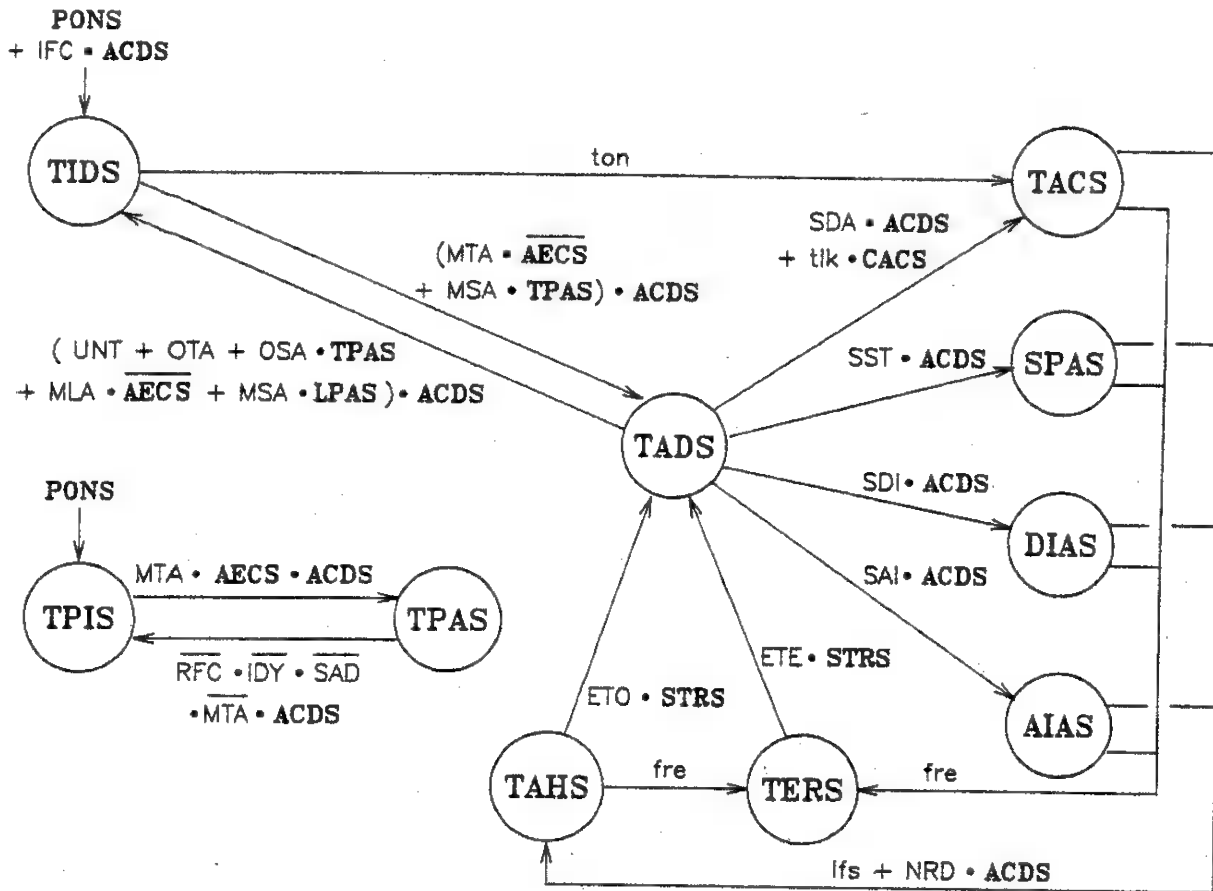
Figure 2-8. T State Diagram

Messages

tlk	local talk	MTA	my talk address	
ton	talk only	NRD	not ready for data	
lfs	last frame sent	OSA	other secondary address	
fre	frame error detected	OTA	other talk address	
		RFC	ready for command	
ETE	end of transmission with error	SAD	secondary address	
ETO	end of transmission, OK	SAI	send accessory ID	
IDY	identify	SDA	send data	
IFC	interface clear	SDI	send device ID	
MLA	my listen address	SST	send status	
MSA	my secondary address	UNT	untalk	

Interface States

AIAS accessory ID active state (links to R,AH,SH)   **ACDS** acceptor data state (from AH)

DIAS device ID active state (links to R,AH,SH)   **AECS** auto extended configured state (from AA)

SPAS serial poll active state (links to R,AH,SH,SR)

TACS talker active state (links to R,AH,SH)   **CACS** controller active state (from C)

TADS talker addressed state (links to R,AH,C,DD)   **LPAS** listener primary active state (from L)

TAHS talker hold state (links to R,AH,SH)

TERS talker error state (links to R,AH,SH)   **PONS** power on state (from PD)

TIDS talker idle state   **STRS** source transfer state (from SH)

TPAS talker primary addressed state (links to R,L)

TPIS talker primary idle state

TIDS (Talker Idle State)

        In this state the device is not able to send data
messages. The function powers on in this state. A talk-only
device may use the local message ton to enter TACS directly from
TIDS without first being addressed. If the MTA (my talk address)
message is received or the MTA and MSA (my secondary address) are
received, and the device is configured for extended addressing,
then the function enters TADS.

TADS (Talker Addressed State)

        In TADS the T function has received its talk address (and
secondary address if extended addressing is used). The function
is prepared for, but not yet engaged in the sending of device
dependent messages over the loop. When the appropriate start of
transmission (SOT) message is received the function enters one of
the active states to begin sending the device dependent messages.
Controllers may use the local message tlk to enter TACS without
receiving the SDA (send data) message. If a CMD which causes the
function to become unaddressed is received while in TADS the
function returns to TIDS.

TACS (Talker Active State)

        In this state the SDA (send data) message has been
received and the device is enabled to begin sourcing device
dependent data messages (DOE). These transmissions are
controlled by the SH (Source Handshake) function. When an NRD
(not ready for data) ready message is received or the device
detects that no more data is ready to be sent, the function
enters TAHS. If the device detects that a DOE frame has been
received in error, then the device enters TERS. If an IFC is
received the function immediately aborts back to TIDS.

SPAS (Serial Poll Active State)

        In this state the SST message has been received and the
device is enabled to replace this message with one or more bytes
of device status information. Transition to TAHS occurs when the
status is sent or when an NRD is received. Transition to TERS
occurs if an error is detected. IFC causes an immediate abort to
TIDS.

DIAS (Device Identify Active State)

        In this state the function has received the SDI message
and the device is enabled to replace this message with its device
ID string followed by a transition to either TAHS or TERS.

AIAS (Accessory Identify Active State)

        In AIAS the SAI frame has been received and the device is
enabled to replace it with one or more bytes of accessory ID
information followed by a transition to either TAHS or TERS.

TAHS (Talker Hold State)

        In talker hold state the device has completed sourcing
its device dependent data and is waiting for the SH (source
handshake) function to indicate that the current handshake cycle
is complete. If an error is detected then the function enters
TERS. If the device does not indicate that an error has
occurred, then ETO (end of transmission, OK) is sourced and the
function enters TADS.

TERS (Talker Error State)

        In TERS the T function has just received the fre local
message indicating that a transmission error has occurred. When
the SH function signals that the current handshake cycle is
complete, the ETE (end of transmission with error) is sourced and
the function enters TADS.

TPIS (Talker Primary Idle State)

        The function is able to respond to its primary talk
address in TPIS but must not recognize its secondary address.
This is the power-on state.

TPAS (Talker Primary Active State)

        In TPAS the function has received its primary talk
address and is now able to recognize and respond to its secondary
address. Receipt of any frame other than IDY, RFC, MSA, or MTA
(again) will cause return to TPIS.

2.9 L (Listener) Function

This interface function provides the device with the capability to receive device dependent data including status, device ID, and accessory ID over the loop from other devices. The L function also allows devices the capability to halt data transfers if enabled to do so by the controller.

When the device is addressed to listen, DOE (data or end) messages received from the loop may be interpreted in a device dependent manner. When not addressed to listen, received data may not be interpreted. Message receiving is controlled by the AH (Acceptor Handshake) interface function. If necessary and enabled by the controller, a listener may halt the data transfer by sending an NRD (not ready for data) sequence to the active talker as controlled by the AH (acceptor handshake) and SH (source handshake) functions.

To enhance the compatibility between the various devices on the loop it is strongly recommended that wherever possible device dependent data messages be sent using the ASCII code.

The talker may send the END message as part of the sent data to indicate an end-of-record condition. The listener(s) may use this information in a device dependent manner. The EOT (end of transmission) message, however, is intended only for the controller and is not interpreted by the listener(s).

The L interface function controls only when data may be accepted from the loop but does not describe how the data is to be interpreted. Since the data transfer may be interrupted so that other operations may be performed, the listener must not allow state changes within the L function to affect the interpretation of the received data.

Listeners may stop active data transfers if enabled to do so by the controller with the ELN (enable listener not ready) command (controllers have this capability intrinsically). If necessary, the device may halt a data transfer with an NRD sequence by sending the local message hlt (halt data transfer) true. If the function is enabled, then it becomes ready to halt the transfer as soon as a DOE (data or end) message is received. When the AH (acceptor handshake) function signals that a DOE

frame has been received and is being held, the NRD message is
sent out on the loop. As soon as the R function indicates that
the NRD has returned, the AH function retransmits the held DOE
frame to the talker. If an IFC (Interface Clear) frame is
received at any time, the function immediately returns to its
inactive state.

        The most important usage of the transfer interrupt
capability is for listeners to halt the input data before a
buffer overrun condition occurs. When a listener detects that it
no longer has room to hold the incoming data, it may send the NRD
message (if enabled) to the talker before it is forced to stop
all loop operations by holding a DOE frame.

        Devices which do not implement the L function are
designated LØ. The basic L capability is designated L1 and
includes LIDS and LACS. If the L function includes the ability
to operate in the listen-only mode, the number 2 is added to the
designation. Devices must allow some means of disabling the lon
message so that it is not always true. Devices without the
listen only capability must always send the lon local message
false. If the device has the capability to use a two byte
address then LIDS, LACS, LPIS, and LPAS are required and the
number 3 is added to the designation. Devices with the
capability to halt data transfers add the number 4 to the
designation and include NIDS, NENS, NRWS, and NACS. For example,
a basic listener with listen-only mode would be written L1,2. A
basic listener with extended addressing capability would be L1,3.
A listener with listen-only capability, extended addressing, and
the capability to halt the data transfer when necessary would be
designated L1,3,4.

$$(\text{MLA} \cdot \overline{\text{AECS}} + \text{MSA} \cdot \text{LPAS}) \cdot \text{ACDS} + \text{lon} + \text{ltn} \cdot \text{CACS}$$

$$\text{lun} \cdot \text{CACS} + (\text{UNL}+\text{IFC}+\text{MTA} \cdot \overline{\text{AECS}} + \text{MSA} \cdot \text{TPAS}) \cdot \text{ACDS}$$

PONS → LIDS ⇄ LACS

PONS → LPIS ⇄ LPAS

$$\text{MLA} \cdot \text{AECS} \cdot \text{ACDS}$$

$$\overline{\text{IDY}} \cdot \overline{\text{RFC}} \cdot \overline{\text{SAD}} \cdot \overline{\text{MLA}} \cdot \text{ACDS}$$

IFC · ACDS

$$\text{ELN} \cdot \text{LACS} \cdot \text{ACDS} + \text{CSBS}$$

PONS → NIDS ⇄ NENS — hlt → NRWS ⇄ NACS

$$\text{CMD} \cdot \overline{\text{ELN}} \cdot \text{ACDS}$$

$$\text{DOE} \cdot \text{DACS} + (\text{CMD}+\text{NRD}) \cdot \text{ACDS}$$

$$\text{DOE} \cdot \text{ACDS}$$

$$\text{NRD} \cdot \text{RCDS}$$
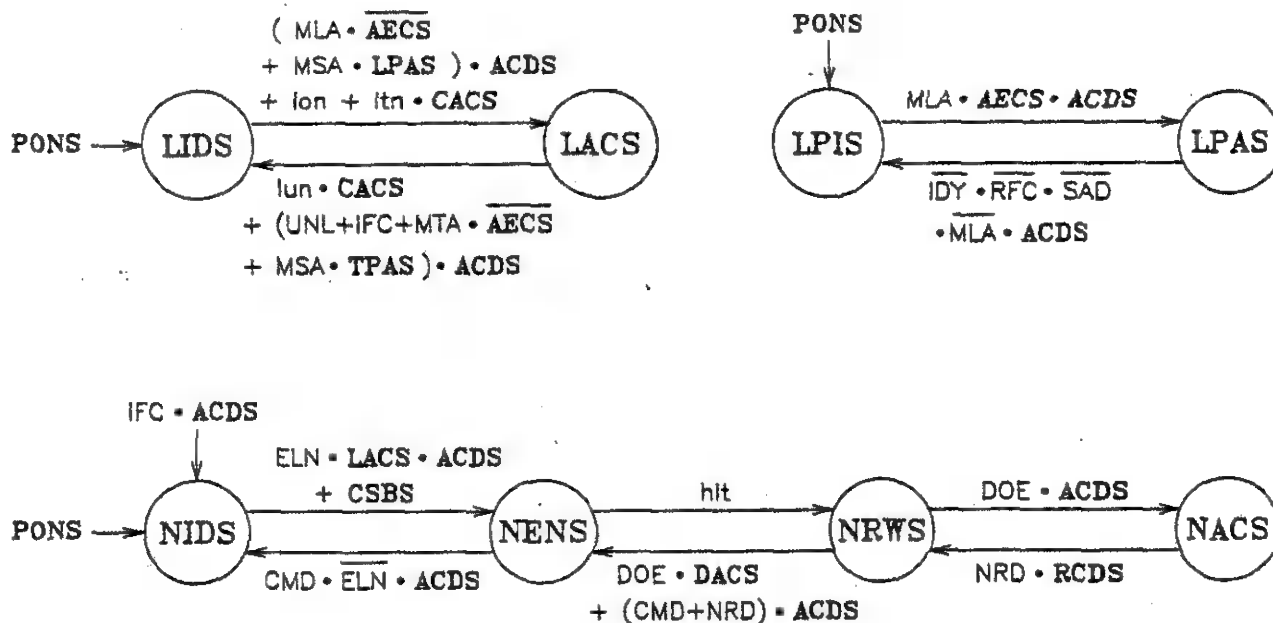
Messages

hlt	halt data transfer		IDY	identify
lon	listen only		IFC	interface clear
ltn	local listen		MLA	my listen address
lun	local unlisten		MSA	my secondary address
			MTA	my talk address
CMD	command		NRD	not ready for data
DOE	data or end		RFC	ready for command
ELN	enable listener not ready		SAD	secondary address
			UNL	unlisten

Interface States

LACS	listener active state (links to R,AH,RL,PP, DC,DT,DD)		ACDS	acceptor data state (from AH)
LIDS	listener idle state		AECS	auto extended configured state (from AA)
NACS	not ready active state (links to AH,SH)		CACS	controller active state (from C)
NENS	not ready enabled state		CSBS	controller standby state (from C)
NIDS	not ready idle state		DACS	driver transmit from acceptor state (from D)
NRWS	not ready wait state (links to R)		PONS	power on state (from PD)
LPAS	listener primary addressed state (links to T)		RCDS	receiver data state (from R)
LPIS	listener primary idle state		TPAS	talker primary addressed state (from T)

Figure 2-9. L State Diagram

)

LIDS (Listener Idle State)

In this state the device is not able to receive device dependent messages. The function powers on in this state. A listen-only device or a controller may use the appropriate local message (lon, ltn respectively) to enter LACS without being addressed to listen. If the MLA message is received and the device is not configured for extended addressing, then the function enters LACS. If the device is configured for extended addressing, both the MLA and MSA (my secondary address) messages must be received before LACS is entered.

LACS (Listener Active State)

)

In LACS the device has received its listen address (and its secondary address if extended addressing is used) and is ready to receive device dependent data messages over the loop. This message transfer is controlled by the AH (Acceptor Handshake) interface function. If a command is received that causes the device to become unaddressed, the function returns to LIDS. A controller device may use the lun local message to cause the function to return to LIDS.

LPIS (Listener Primary Idle State)

In LPIS the device is able to recognize its primary listen address but is not able to respond to its secondary address. The L function powers on in this state. If the device is configured for extended addressing, then the MLA (my listen address) message will cause a transition to LPAS.

LPAS (Listener Primary Active State)

The function has received its primary listen address and is now able to respond to and recognize its secondary address in this state. The receipt of any frame other than MSA, MLA (again), RFC, or IDY will cause a return to LPIS.

2-34

)

NIDS (Not Ready Idle State)

       This is the power-on state. In NIDS the function is not
enabled to interrupt a data transfer. If the device is active
listener and the ELN (enable listener not ready) command is
received or the device is controller, the function enters NENS.

NENS (Not Ready Enabled State)

       In NENS the function has been enabled to perform the NRD
sequence but is not actively doing so. If the device decides
that the data transfer should be halted, it sends the local
message hlt true and the function enters NRWS. If any command is
received, the function returns to NIDS.

NRWS (Not Ready Wait State)

       In this state, the function is waiting for a data byte to
be received so that it can perform the NRD sequence. When a DOE
frame is received, the function enters NACS. If a CMD or NRD
frame is received, the function returns to NENS.

NACS (Not Ready Active State)

       In NACS the function is in the process of sourcing the
NRD frame. When the NRD frame returns, the function returns to
NRWS.

2.10 SR (Service Request) Function

The SR function provides the device the ability to request service from the controller. Devices may request service by setting the SRQ (CØ) bit in DOE and IDY messages. The set SRQ bit notifies the controller that one or more devices on the loop have requested service. CMD and RDY frames do not have an SRQ bit and may not transmit the service request message.

The device enables the SR function to send the service request message by setting the local message rsv true. When the controller asks the device to send its status, the SR function stops sending the SRQ message on the loop unless the local message rsv is set false and then true again.

A device may send the service request message to the controller by setting the SRQ bit on any DOE or IDY frame that the device sources or retransmits. This requires a commitment on the part of the controller to source IDY frames periodically when no data is being transferred to check for service requests. If the loop is to be idle and periodic polling is not desirable, the controller may enable devices to source asynchronous IDY frames with the SRQ bit set by sending the EAR (enable asynchronous requests) command. When the controller receives an asynchronous IDY or decides on its own that it needs to perform further operations, it will source a UCG (universal command) to disable further asynchronous requests and proceed with normal operations. While devices are enabled to source asynchronous IDYs, the controller must not send any frames except UCG or IDY. Since asynchronous IDYs may not travel the entire loop, the parallel poll bits will not necessarily be correct.

Two bits in the first byte of a device's status are reserved and must not be used for any other purpose. Bit 7 (msb) is set to indicate that the first byte is a system status message and is clear otherwise. System status messages are a device independent method of communicating status information (Refer to appendix C for a complete list of system status messages and their usage). Bit 6 is always equal to the device's local message rsv.

To enhance the compatibility between many different devices, system status messages should be used whenever possible. This allows generalized controllers to be implemented that can understand status messages from any device on the loop. Most devices can describe all possible status with the system messages defined. If more specific information is necessary or desirable, the device may send additional bytes of status.

Devices that do not implement system status should not request service for reasons other than data integrity unless enabled by the controller. Events which involve data integrity are defined to be those events that risk either data loss or an infinite halt of the loop data transfer. The controller may enable devices to send the SRQ message for causes other than data integrity (if the device has the capability) using either device dependent commands or ASCII commands defined for that purpose.

Devices that implement system status should request service for data integrity reasons and may optionally send the SRQ message whenever the current status changes. System status messages are required for change of status SRQs because the controller needs no device specific information to correctly interpret the request.

When a controller receives a service request from a device, it should stop the current operation immediately and find the reason for the SRQ. If data loss is a possibility, there may be a limited amount of time before data is actually lost.

SRØ indicates no service request capability. Basic service request capability is indicated by SR1 with the states ARSS, ARIS, and ARAS omitted. SR2 indicates full SR capability including asynchronous service requests with no states omitted. SR1 and SR2 both require the T2 subset of the talker function to send the device's status byte(s). Note that the arq message may be sent true only when SRSS is active. It must be sent false immmediately on entry to ARSS.
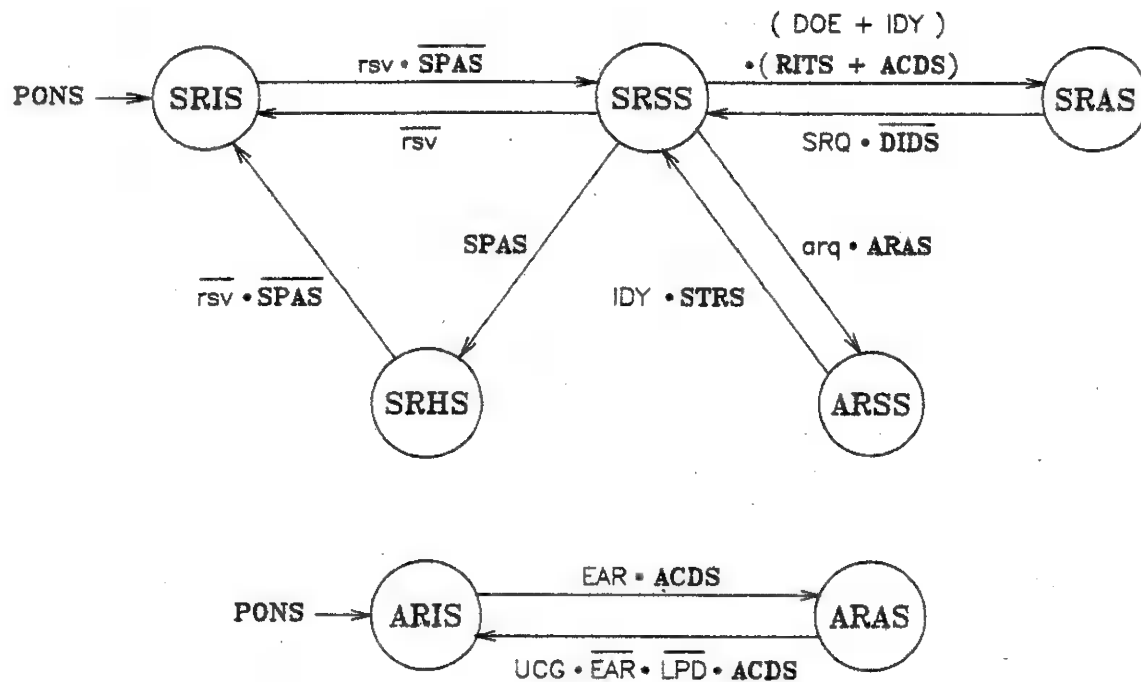
Messages

rsv	request service		IDY	identify
arq	asynchronous request		LPD	loop power down
			SRQ	service request
DOE	data or end		UCG	universal command
EAR	enable asynchronous requests			

Interface States

ARAS	asynchronous request active state		ACDS	acceptor data state (from AH)
ARIS	asynchronous request idle state		DIDS	driver idle state (from D)
ARSS	asynchronous request source state (links to SH)		PONS	power on state (from PD)
			RITS	receiver immediate transfer state (from R)
SRIS	service request idle state			
SRSS	service request standby state		SPAS	serial poll active state (from T)
SRAS	service request active state		STRS	source transfer state (from SH)
SRHS	service request hold state			

Figure 2-10.   SR State Diagram

SRIS (Service Request Idle State)

        In SRIS the SR function is not requesting service. This
is the power-on state. When the rsv local message goes true the
function enters SRSS provided the controller is not currently
requesting the device's status byte(s) via a serial poll.

SRSS (Service Request Standby State)

        In this state the device is waiting for an opportunity to
send the SRQ message. When a DOE or IDY frame arrives the
function enters SRAS. If ARAS is active and the arq local
message is true the function enters ARSS. If the device no
longer requires service the function returns to SRIS. When the
controller conducts a serial poll to retrieve the device's status
the function enters SRHS.

SRAS (Service Request Active State)

        In SRAS a DOE or IDY message has been received and the
device is setting the SRQ bit in this message prior to its
transmission by the D (Driver) interface function. When the
transmission begins, the function returns to SRSS.

ARSS (Asynchronous Request Source State)

        In this state the device sources its own IDY message with
the SRQ bit set. When the D function indicates that transmission
has begun the function returns to SRSS.

SRHS (Service Request Hold State)

        In SRHS the device requires service but may not request
it over the loop. When the device sends the local message rsv
false, the function returns to SRIS.

ARIS (Asynchronous Request Idle State)

        In ARIS the device may not generate asynchronous service
requests. This is the power on state. When the EAR message is
received the function moves to ARAS.

ARAS (Asynchronous Request Active State)

        In this state the device is enabled to generate
asynchronous service requests. If any universal command other
than EAR or LPD is received the function returns to ARIS.

)

2.11 RL (Remote Local) Function

The RL interface function provides a device with the capability to select between two sources of input information. The input may either be from a local source (such as a front panel keyboard) or from the interface.

The RL function allows the device to have three basic levels of local control. Full local control allows all local controls such as switches or keyboards to be fully operative. Device commands received in this mode may be ignored or treated as device dependent data. Remote control requires the device to ignore all local controls that have corresponding remote controls. Local controls that generate local messages to the interface functions will continue to be operative including the rtl message which can be used to return the function to local control. Remote control with local lockout requires the device to ignore all local controls including those that generate local messages.

Note that this function does not affect a device's ability to send or receive messages on the loop, although device interpretation of data messages may be altered. Selected devices may be placed in remote control by sending the device's listen address, while others are left under local control. Note also that the IFC (Interface Clear) message does not affect the RL function.

When the function transitions from one of the local states to one of the remote states the device functions may retain their local settings until receiving specific remote settings, or they may immmediately revert to the remote settings previously received. Conversely, when returning to local control, the device functions may retain their remote settings until local controls override them, or they may immediately revert to the present settings of the local controls.

Devices without RL capability are designated RLØ. RL1 indicates basic RL capability without local lockout and includes all states except LWLS and RWLS. Note that devices designated RL1 must always send the rtl local message false. RL2 indicates full RL capability including local lockout and includes all states. Both RL1 and RL2 designations require the L function.

)
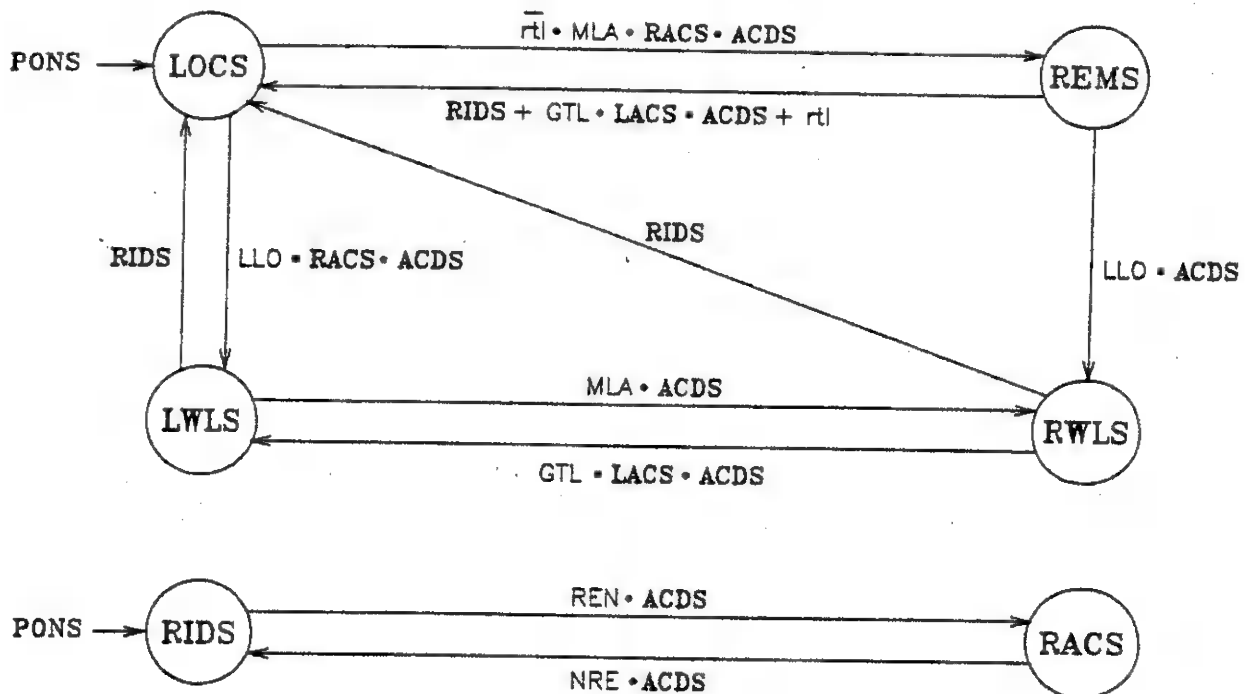
Messages

rtl	return to local		MLA	my listen address
			NRE	not remote enable
GTL	go to local		REN	remote enable
LLO	local lockout			

Interface States

LOCS	local state		ACDS	acceptor data state (from AH)
LWLS	local with lockout state		LACS	listener active state (from L)
RACS	remote active state		PONS	power on state (from PD)
REMS	remote state			
RIDS	remote idle state			
RWLS	remote with lockout state			

Figure 2-11.   RL State Diagram

2-41

RIDS (Remote Idle State)

In RIDS the function is not enabled for remote operation.
The function powers on in this state. If the REN (remote enable)
message is received the function enters RACS.

RACS (Remote Active State)

In this state the function is indicating that the active
controller has enabled the interface for remote operation,
however, the device functions remain under local control if LOCS
is active. If the NRE (not remote enable) message is received
the function immediately returns to LOCS and RIDS.

LOCS (Local State)

In LOCS all local controls of the associated device
functions (both front and rear panel) are operative. The device
may store but not respond to corresponding device dependent
messages from the interface. The RL function powers on in this
state. If RACS is active and MLA is received the function will
enter REMS, provided the rtl local message is not true. If LLO
(local lockout) is received the function enters LWLS.

LWLS (Local With Lockout State)

In this state, as in LOCS, the local controls are
operative. The device may store but not respond to corresponding
device dependent messages from the loop. If RACS is active and
MLA is received the function enters RWLS. NRE causes the
function to return to LOCS.

REMS (Remote State)

In REMS the local controls which have corresponding
remote controls are inoperative and those device functions are
under control of messages received from the interface. This does
not include those controls which generate local messages for the
interface functions. If the NRE or GTL messages are received or
if the local message rtl goes true the function returns to LOCS.
If the LLO message is received the function enters RWLS.

RWLS (Remote With Lockout State)

In this state all local controls are inoperative and the
function will not respond to the rtl local message. If the GTL
message is received the function goes to LWLS. If the NRE
command is received, the function will return to LOCS.

2.12 AA (Automatic Address) Function

The AA function gives the device the ability to have its address assigned by the active controller. Either a one byte or a two byte address may be assigned to each device. If all devices have the capability to accept a two byte address, a maximum of 961 devices may be on the loop.

At power-on the device has no address assigned to it and may elect to respond to a default address or respond to no address at all. When an implemented addressing sequence is received, the device becomes configured for the assigned address. While the device is configured, it may not respond to any other address or accept another address assignment. If at any time the AAU (auto address unconfigure) message is received, the function immediately returns to its unconfigured state and the device must assign itself a default or switch selectable address. Note that since devices may not have any address at all at power-on, controllers intending to use the default or switch selectable addresses must send the AAU command at least once.

The least significant five bits of the AAD (auto address) AEP (auto extended primary), AES (auto extended secondary), and AMP (auto multiple primary) messages contain the binary address assignment, addresses 0 through 30. Address 31 is not a legal address (11111). This message is called IAA, IEP, IES, or IMP (illegal AAD, AEP, AES, AMP respectively) and the AA function does not respond in any way to these message though it might generate them in response to an incoming auto address message.

To configure the loop for one byte addressing, the controller would send the AAD1 (auto address 1) message (reserving address 0 for itself). If a valid AAD message returns, then the address it contains is one more than the number of devices on the loop. If the returned message is IAA then the loop may contain more than 30 devices and the controller should send AAD30 to check. If the AAD30 returns unchanged then there are exactly 30 devices on the loop and the controller may proceed as usual. If IAA is returned again, more than 30 devices are on the loop and improper operations may result.

Auto extended addressing allows each device on the loop to be asigned a two byte address so that a maximum of 961 devices

may be connected at once. For configuration, the controller
would send the AES1 (auto extended secondary 1) message
(reserving address 0,0 for itself). The first group of devices
would receive their secondary addresses and increment the AES
until IES is generated. When the controller receives the IES
message, it will send the AEP0 (auto extended primary 0) message.
Only those devices which have received legal secondary addresses
will respond to and accept this primary address. This group is
now configured and will not respond to any auto address messages
until AAU is received. Note that the primary address is not
incremented by the loop devices and must be incremented by the
controller. The controller should now send AES0 followed by AEP1
to configure the next group of devices. The controller should
continue this AES0-AEPn sequence until the AES message returns
with a legal address (not IES). The value of the returned AES
message and the coresponding AEP message sent indicate how many
devices are on the loop. This addressing sequence must be sent
before the auto multiple sequence if both are to be used on one
loop, or it will not work correctly.

        Auto multiple addressing allows devices with multiple
functions to be assigned one primary address and have a separate
secondary address for each function. This form of addressing
allows as many as 31 devices (including the controller) with up
to 31 functions in each. The configuration sequence begins when
the controller sends the AMP1 (auto multiple primary) message
(reserving address 0 for itself). The address returned in the
AMP message indicates the number of devices connected (IMP,
illegal multiple primary, would indicate that there may be more
than 30 devices on the loop as in simple auto addressing). Then
the controller would send ZES (zero extended secondary) once for
each device. The return values indicate the number of separate
functions in each device. If both auto extended and auto
multiple addressing are to be implemented in the same loop, the
auto extended addressing sequence must be sent first so that the
ZES message of the auto multiple sequence is not incorrectly used
by extended addressing devices

        Simple auto addressing is designated AA1. Extended
addressing is designated AA2. Auto multiple addressing is
designated AA3. All devices must implement one or more of these
three subsets. For example, a device that can be assigned either
one or two byte addresses is designated AA1,2.
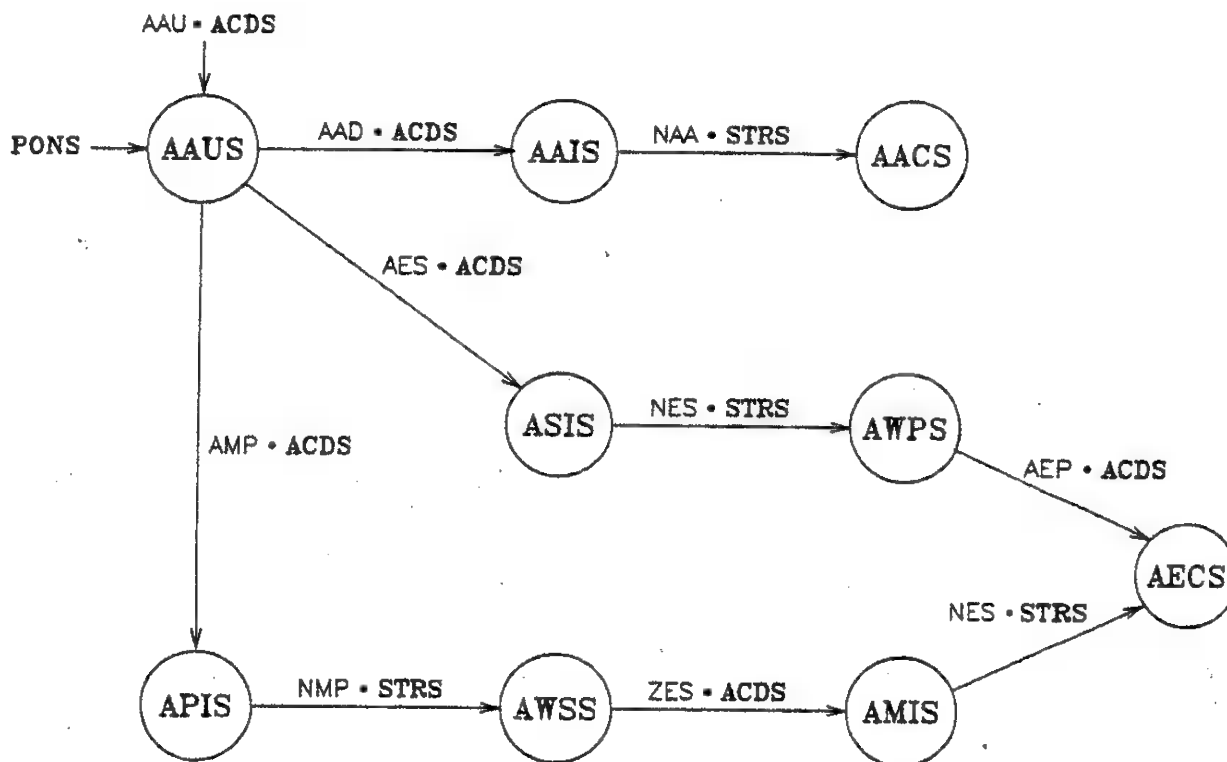
Messages

AAD	auto address		NAA	next auto address
AAU	auto address unconfigure		NES	next extended secondary
AEP	auto extended primary		NMP	next multiple primary
AES	auto extended secondary		ZES	zero extended secondary
AMP	auto multiple primary			

Interface States

AACS	auto address configured state		ACDS	acceptor data state (from AH)
AAIS	auto address increment state (links to SH)		PONS	power on state (from PD)
AAUS	auto address unconfigured state (links to R)		STRS	source transfer state (from SH)
AECS	auto extended configured state (links to L,T)			
AMIS	auto multiple increment state (links to SH)			
APIS	auto primary increment state (links to SH)			
ASIS	auto secondary increment state (links to SH)			
AWPS	auto wait for primary state (links to R)			
AWSS	auto wait for secondary state (links to R)			

Figure 2-12.   AA State Diagram

AAUS (Auto Address Unconfigured State)

        In AAUS the function has no automatic address assignment.
At power-on the device may elect to respond either to a preset or
switch selectable address or to no address at all (in order to
reduce the chances of two devices having the same address).
After the AAU message is received, the device must respond to
either a default or switch selectable address. If the AAD (auto
address), AES (auto extended secondary), or AMP (auto multiple
primary) messages are received, the function enters AAIS, ASIS,
or APIS respectively. AAUS is the power-on state.

AAIS (Auto Address Increment State)

        In AAIS the AAD message has been received with the
device's primary address contained in the lower five bits. In
this state the device accepts the address assignment and then
increments it to generate the NAA (next auto address) message for
the next device. When the NAA transmission begins the function
enters AACS.

AACS (Auto Address Configured State)

        In this state the device is configured for one byte
addressing and must only respond to the address assigned by the
active controller in the AAD message. The device may not respond
to AAG (auto address group) messages while AACS is active. If
the AAU command is received the function returns to AAUS.

ASIS (Auto Secondary Increment State)

        In ASIS the AES message has been received with the
device's secondary address assignment contained in the lower five
bits. The device accepts this assignment and increments the AES
message to generate the NES (next extended secondary) message for
the next device. When transmission of NES begins the function
enters AWPS.

AWPS (Auto Wait for Primary State)

        In AWPS the function has received its secondary address
assignment and has sent the NES message to the next device. The
function now waits for the AEP message containing its primary
address. The function enters AECS when AEP is received. The AAU
command causes a transition to AAUS.

APIS (Auto Primary Increment State)

        In APIS the function has received the AMP message
containing its primary address assignment in the lower five bits.
The function accepts the assigned primary address and increments
it to generate the NMP (next multiple primary) message for the
next multiple function device. When transmission of NMP begins
the function enters AWSS.

AWSS (Auto Wait for Secondary State)

        The function has its primary address and is waiting for
the ZES (zero extended secondary) message. When the ZES message
is received, the function enters AMIS. If AAU is received the
function returns to AAUS.

AMIS (Auto Multiple Increment State)

        In AMIS the device has received the ZES message and is
assigning each function of the device a secondary address
starting with zero. The last address assigned within the device
becomes the NES message. Note that all devices to be configured
for extended addressing (and not auto multiple addressing) must
already be configured or they will respond to the NES message.
When transmission begins, the function enters AECS.

AECS (Auto Extended Configured State)

        In this state the device is configured for two byte
addressing and must only respond to the primary and secondary
addresses assigned by the active controller. The function will
no longer respond to any remote messages except AAU. If AAU is
received the function returns to AAUS.

2.13 PD (Power Down) Function

This function gives a device the capability to be placed
in a power-down or low power mode by command received over the
loop and likewise to be reactivated for normal operation. When
in the power-down mode the device only needs to monitor the loop
for any pulse. When any frame arrives the edge pseudomessage is
generated which reactivates the device and places the interface
functions in the power-on state.

If the PD function receives the LPD (loop power down)
command it will proceed to turn the device off after the other
interface functions have signaled that they are ready. If the
local message fon (force on) is true, the device does not power
down, but instead returns to the normal power-up state. If an
edge is detected while the device is powered down, the function
initializes the interface and device functions and returns to the
normal power-up state. The local message pof allows a device to
be powered down without receiving a command from the loop. If an
edge is detected while the pof message is true, the device will
remain in the powered-off state.

The active controller should wake up a powered-down loop
by repeatedly sending a universal command (NOP is recommended)
until it returns followed by one and only one RFC to complete the
handshake. This will permit devices whatever time is necessary
to complete their individual power-on sequences and begin to
recognize and retransmit frames. The controller would do this
either on its own initiative or in response to an asynchronous
IDY if this mode was enabled prior to power-down.

Note that devices must not retransmit any frames after
the LPD and RFC until the power down operation is complete.
Devices may use the local message fon to disable the power down
function, if desired. Because it must be able to wake up a
powered down loop, the controller may not allow its interface
functions to be powered down.

All devices must implement at least POFS, PONS, and PUPS.
This capability is designated PD0 and requires that the edge
message is always false. PD0 devices simply ignore the LPD
command. PD1 indicates full capability. All states must be
implemented as shown. No other subsets are permitted.

Figure 2-13. PD State Diagram
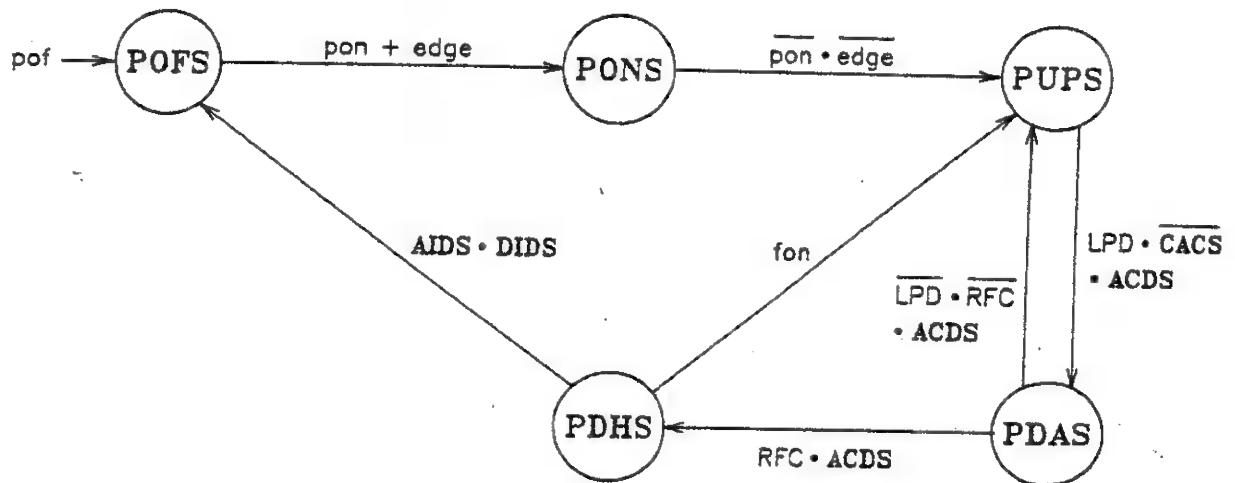
Messages

edge	pulse edge detected		LPD	loop power down
fon	force on		RFC	ready for command
pof	power off			
pon	power on			

Interface States

PDAS	power down active state		ACDS	acceptor data state (from AH)
PDHS	power down hold state		AIDS	acceptor idle state (from AH)
PONS	power on state (links to all		CACS	controller active state (from C)
	interface functions)		DIDS	driver idle state (from D)
POFS	power off state			
PUPS	power up state			

POFS (Power Off State)

        This state is entered whenever the local message pof is
true. This message is normally generated from the off position
of the device power switch. All of the other interface functions
have the equivalent of this state although it is only shown here
for clarity. When POFS becomes active in the PD function, it
causes an immediate transition to this equivalent state in all
other interface functions. When the device power switch
generates the pon local message or a frame arrives generating the
edge message, the function enters PONS.

PONS (Power On State)

        PONS causes the entry to the first (power-on) state in
all other interface functions. The pon or edge message must
remain true for sufficient time to allow interface and device
functions to properly respond. When these messages both are
false again, the function enters PUPS, the normal operating
state.

PUPS (Power Up State)

        PUPS is the normal operating state for the device. If
the LPD command is received and the device is not the active
controller, the function enters PDAS. The active controller must
remain on in order to power up the loop when necessary. If the
controller is also the system controller, it may allow itself to
power down.

PDAS (Power Down Active State)

        In PDAS the device is waiting for the RFC following the
LPD command. When the RFC is received the function enters PDHS.
If any other remote message is received the function aborts and
returns to PUPS.

PDHS (Power Down Hold State)

        In this state the function is waiting for the completion
of the retransmission of the RFC message before actually going to
the power down mode, POFS. If the fon local message is true, the
function will return to PUPS.

2.14 PP (Parallel Poll) Function

The PP interface function gives a device the ability to present to the controller one bit of status information without being addressed to talk. The eight data bits in the IDY message carry this status information from the devices to the controller. Any number of devices can be accomodated by sharing the available bits. Devices can be configured to present the logical OR or the logical AND of their respective status on a particular bit.

Whenever a device receives the PPE (parallel poll enable) command while addressed to listen, it becomes enabled to respond to parallel polls on the assigned bit with the assigned polarity on any IDY message received. This may result in setting one bit of an IDY frame before retransmission. The controller may assign a new response bit or polarity to a device at any time by sending a new PPE command while the device active listener. The PPD (parallel poll disable) addressed command and the PPU (parallel poll unconfigure) universal commands disable the parallel poll function.

An important use of parallel poll is reducing the controller's service request response time. When the SRQ message is received by the controller, a parallel poll may be conducted first to narrow the search. If the devices of interest are appropriately configured for parallel poll, the device requiring service may be located after sourcing only a few frames.

Three bits (D2, D1, and D0) in the PPE command represent the binary bit number on which to respond to parallel polls (000 assigns bit D0 (lsb), 001 indicates bit D1, etc.). Another bit in the PPE command (D3) indicates the logical sense of a true response. If bit D3 is a 0, the device's positive response is to not modify the incoming bit of the IDY; the negative response would be to OR in a 1 in the assigned bit. If bit D3 is a 1, the positive response is to OR in a 1, the negative response is no modification. This permits devices to be configured to present the logical OR or the logical AND of their responses. Note that regardless of the logical sense assigned, the device may only OR its response to the assigned bit and therefore may not affect the bit if it is received as a 1.

No capability is designated PP0. Full capability is designated PP1. No subsets are permitted.

2-51

PONS → (PPIS) ⟶ PPE • LACS • ACDS ⟶ (PPSS) ⟶ IDY • RITS ⟶ (PPAS)

(PPU+PPD • LACS) • ACDS   IDY • DTRS
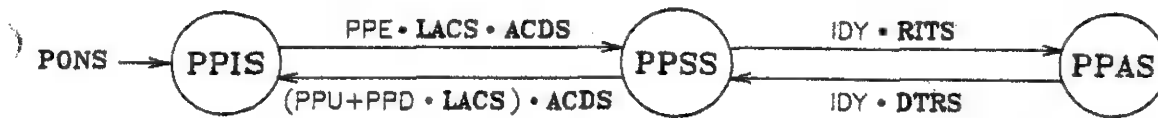
Messages

IDY identify
PPD parallel poll disable

PPE parallel poll enable
PPU parallel poll unconfigure

Interface States

PPAS parallel poll active state
PPIS parallel poll idle state
PPSS parallel poll standby state

ACDS acceptor data state (from AH)
DTRS driver transfer state (from D)
LACS listener active state (from L)
PONS power on state (from PD)
RITS receiver immediate transfer state
(from R)

Figure 2-14. PP State Diagram

PPIS (Parallel Poll Idle State)

PPIS is the power-on state. In PPIS, the PP function
may not respond to parallel polls and must retransmit all IDY
messages exactly as received. When the PPE command is received,
the function enters PPSS.

PPSS (Parallel Poll Standby State)

In PPSS the device is configured to respond to parallel
polls and is waiting for the active controller to source the IDY
message. When an IDY frame is received, the function enters
PPAS. If a new PPE command is received, the function accepts a
new bit number and polarity for PP response.

PPAS (Parallel Poll Active State)

In PPAS the function is actively responding to a parallel
poll. The device may modify the IDY frame before retransmission
by setting the assigned bit according to the assigned sense and
the device's individual status. When the D function signals
that retransmission has begun, the function returns to PPSS.

2.15 DC (Device Clear) Function

The DC interface function provides the device with the ability to be cleared or initialized. The Controller may choose to send the clear message to all devices on the loop or only to those devices addressed to listen.

Through the DC function, the device may receive and respond to the DCL (device clear) and SDC (selected device clear) messages. The DCL or SDC message allows the device to alter device conditions or states usually returning them to their initial or power-on states. The device may not alter any interface functions in response to the clear message. If both the DCL message and the SDC message are implemented they must affect exactly the same device functions in exactly the same way.

Note that this function affects only the device function and has no effect on the other interface functions (which are initialized by the IFC message or the power on state). The device may use the DC function for any purpose consistent with its operation. This would typically include placing the device (not the interface) in its power-on state, but may be used to place the device function in any predefined state as specified by the designer.

If the DC function is not implemented the designation is DC0. DC1 designates the ability to respond to the DCL message. All devices with DC1 are cleared together by the DCL message. If the device also includes the ability to respond to the SDC message DC2 is the designation. This message clears only addressed devices. This capability requires the L function and the optional terms in the DC state diagram. If this capability is not implemented the optional terms must be omitted. Repeating, a device which can only respond to DCL is designated DC1 while a device that can respond to both DCL and SDC is designated DC2.

$$(\text{DCL} \left[+ \text{SDC} \cdot \text{LACS} \right]) \cdot \text{ACDS}$$

PONS → DCIS ⇄ DCAS
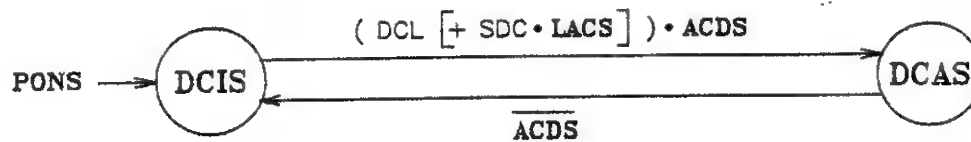
$$\overline{\text{ACDS}}$$

Messages

DCL   device clear   SDC   selected device clear

Interface States

DCAS   device clear active state   **ACDS**   acceptor data state (from AH)
DCIS   device clear idle state   **LACS**   listener active state (from L)
                                              PONS   power on state (from PD)

Figure 2-15.   DC State Diagram

DCIS (Device Clear Idle State)

In this state the DC function is not actively clearing the device functions. When the DCL message is received or the SDC message is received and the device is in Listener Active State, the function enters DCAS.

DCAS (Device Clear Active State)

In DCAS the the function is signaling the device to clear or initialize its internal function. As soon as the device signals to the AH function that it has accepted the clear message, the function returns to DCIS.

2.16 DT (Device Trigger) Function

The DT function gives the device the ability to have its basic operation started on command. Since the device must be addressed to listen in order to respond to the trigger command, either a single device or a group of devices may be triggered at the same time. The command does not affect other interface functions.

Through the DT function, the device may receive and respond to the GET (group execute trigger) message. To trigger a device to begin its basic operation, the active controller must source the GET command when the device is active listener. An important use of the DT function is to allow the controller to synchronize the basic operations of several devices on the loop. Therefore, it is recommended that the device commence its basic operation as soon as DTAS becomes active. Once the device operation is started the device should not respond to further transitions of the DT interface function until the operation is complete.

Note that the GET message affects only the device functions and has no effect on any other interface functions. When the trigger message is received, the device may perform any action consistent with the device's operation as defined by the device designer.

If the device does not include the device trigger capability it is designated DT0. If the complete device trigger capability is present, it is designated DT1 and requires the L function. No subsets are permitted.
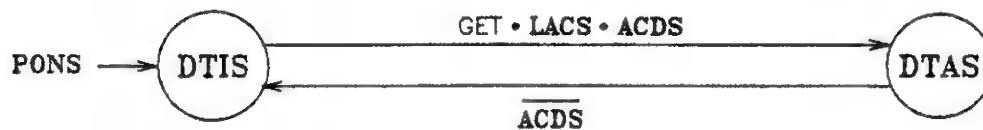
Messages

GET   group execute trigger

Interface States

DTAS   device trigger active state   ACDS   acceptor data state (from AH)
DTIS   device trigger idle state   LACS   listener active state (from L)
                                             PONS   power on state (from PD)

Figure 2-16.   DT State Diagram

DTIS (Device Trigger Idle State)

        In this state the DT function is not active. The
junction powers on in this state. If the GET command is received
and the device is listener active, the function enters DTAS.

DTAS (Device Trigger Active State)

        In DTAS the DT function is signaling the device function
to start performing its basic operation. When the device
indicates to the AH (acceptor handshake) function that the trigger
message has been received, the function returns to DTIS.

2.17 DD (Device Dependent Command) Function

The DD function allows the device to accept device dependent commands from the interface to be used as the designer chooses. Sixty-four command codes are reserved for device dependent purposes. There are 32 DDL commands that a device may interpret only if it is active listener. There are also 32 DDT commands that a device may interpret only if addressed to talk.

The device dependent commands received by the DD interface function must affect only device functions. Interface functions such as talker and listener must not be affected by any device dependent command.

It is recommended that the device commence its response as soon as DDAS becomes active. Once this response is begun the device should not respond to further transitions of the DD interface function until the operation is complete.

A separate DD interface function should be implemented for each device dependent command to which the device responds. Only the appropriate one of the two optional terms in the transition will be implemented in any particular DD function.

No capability is designated DD0. If the device responds to one or more DDL or DDT commands, it is designated DD1. DD1 devices must include appropriate documentation on each device dependent command response.

$$[\text{DDLn} \cdot \text{LACS} \cdot \text{ACDS}] + [\text{DDTn} \cdot \text{TADS} \cdot \text{ACDS}]$$

PONS $\longrightarrow$ DDIS $\xrightarrow{\hspace{4cm}}$ DDAS
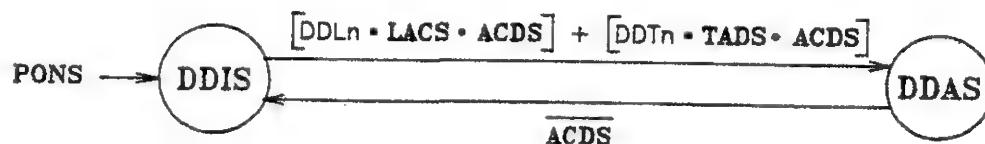
$\overline{\text{ACDS}}$

Messages

DDLn device dependent listener command n   DDTn device dependent talker command n

Interface States

DDAS device dependent active state   **ACDS** acceptor data state (from AH)
DDIS device dependent idle state   **LACS** listener active state (from L)
                                                 PONS power on state (from PD)
                                                 TADS talker addressed state (from T)

Figure 2-17. DD State Diagram

DDIS (Device Dependent Idle State)

        In this state the device is not performing the operation
the designer has specified for the command represented by this
interface function. When the command is received and the device
is appropriately addressed, the function transitions to DDAS.

DDAS (Device Dependent Active State)

        In DDAS the function is indicating to the device that it
should begin performing the operation specified for the command
represented by this interface function.

2.18 Remote Message Coding

This section gives definitions and codes for all remote messages. Remote messages on the signal lines are sensed by the decoder, are decoded, and are passed on to the interface functions for appropriate action. Interface functions originate and control the transmission of remote messages to the encoder, which encodes and transmits them over the signal lines.

Throughout this section, Ø represents a logical zero signal, 1 represents a logical one signal, and X represents a receiver "don't care" or a transmitter "no change" for retransmission. The letters a, b, m, n, s, and t are used to describe bits with specific meanings. These are described in the notes at the end of the section.

The first three bits of each frame (C2, C1, CØ in that order) are called control bits and serve to classify frames into major categories for retransmission and response. The following table gives the codes for these bits of the frame and their meanings. Please note that the interface system responds in precisely the same way to both data and end messages. The end bit is provided solely as a mechanism for devices to use to indicate an end-of-record condition within device dependent message transmissions without necessarily terminating the transmission. The end bit is C1 and the service request bit is CØ in those frames which have provision for these bits.

CONTROL BIT CODES

C2	C1	CØ	Classification	Mnemonic
Ø	Ø	Ø	data byte	DAB
Ø	Ø	1	data byte with service request	DAB (SRQ)
Ø	1	Ø	end byte	END
Ø	1	1	end byte with service request	END (SRQ)
1	Ø	Ø	command	CMD
1	Ø	1	ready	RDY
1	1	Ø	identify	IDY
1	1	1	identify with service request	IDY (SRQ)

)   The remaining eight bits (D7 through DØ in order) are
called data bits (not to be confused with a data message) and
indicate the specific message within the general classifications
described above. The specific codes for command, ready, and
identify messages are given in this section. The codes for data
or end messages may be any eight bit pattern which the involved
devices are all able to interpret. It is recommended for maximum
compatibility, however, that the ASCII code be used whenever
possible. Bit Ø of the ASCII code representation corresponds to
HP-IL bit DØ and bit 7 corresponds to D7. Regardless of the data
coding used, it is recommended that the most significant bit be
sent on D7 and the least significant bit correspond to DØ.

        Messages may be defined as the logical combination (AND,
OR, or NOT) of other messages. Messages are therefore divided
into several classes and sub-classes. To facilitate the user's
understanding of these message relationships a frame hierarchy
table is included. This table graphically shows which messages
are included within other messages. An alphabetic listing of all
messages, their mnemonics, and their coding follows the frame
hierarchy table. For convenience, a table of messages by coding
is also included.

)

)

FRAME HIERARCHY TABLE

```
DOE .... DAB                          IDY .... IDY
         . DAB(SRQ)                           . IDY(SRQ)
         . END
         . END(SRQ)                   RDY .... RFC
                                              .
CMD .... ACG .... NUL                         . ARG .... EOT .... ETO
         .        . GTL                       .        .        . ETE
         .        . SDC                       .        .
         .        . PPD                       .        . NRD
         .        . GET                       .        .
         .        . ELN                       .        . SOT .... SDA
         .        . PPE0(0-7)                  .        .        . SST
         .        . PPE1(0-7)                  .        .        . SDI
         .        . DDL(0-31)                  .        .        . SAI
         .        . DDT(0-31)                  .        .        . TCT
         .                                     .
         . UCG .... NOP                        .
         .        . LLO                        . AAG .... AAD(0-30)
         .        . DCL                                 . NAA(1-31)
         .        . PPU                                 . IAA
         .        . EAR                                 .
         .        . IFC                                 . AEP(0-30)
         .        . REN                                 . IEP
         .        . NRE                                 .
         .        . AAU                                 . ZES
         .        . LPD                                 . AES(0-30)
         .                                              . NES(1-31)
         . LAG .... LAD(0-30)                            . IES
         .        . MLA(0-30)                            .
         .        . UNL                                  . AMP(0-31)
         .                                               . NMP(1-31)
         . TAG .... TAD(0-30)                            . IMP
         .        . MTA(0-30)
         .        . OTA(0-30)
         .        . UNT
         .
         . SAG .... SAD(0-30)
                  . MSA(0-30)
                  . OSA(0-30)
```

MESSAGE TABLE

MNEMONIC	MESSAGE	CLASS	GROUP	CCC 210	DDDDDDDD 76543210	NOTES
AAD	auto address 0-30	RDY	AAG	101	100aaaaa	1
AAG	auto address group	RDY		101	1xxxxxxx	
AAU	auto address unconfigure	CMD	UCG	100	10011010	
ACG	addressed command group	CMD		100	x000xxxx	
			or	100	101xxxxx	
			or	100	110xxxxx	
AEP	auto extended primary 0-30	RDY	AAG	101	101aaaaa	1
AES	auto extended secondary 0-30	RDY	AAG	101	110aaaaa	1
AMP	auto multiple primary 0-30	RDY	AAG	101	111aaaaa	1
ARG	addressed ready group	RDY		101	01xxxxxx	
CMD	command			100	xxxxxxxx	
DAB	data byte	DOE		00x	xxxxxxxx	
DCL	device clear	CMD	UCG	100	00010100	
DDL	device dependent listener command 0-31	CMD	ACG	100	101xxxxx	
DDT	device dependent talker command 0-31	CMD	ACG	100	110xxxxx	
DOE	data or end			0xx	xxxxxxxx	
EAR	enable asynchronous requests	CMD	UCG	100	00011000	
ELN	enable listener not ready	CMD	ACG	100	00001111	
END	end	DOE		01x	xxxxxxxx	
EOT	end of transmission	RDY	ARG	101	0100000x	
ETE	end of transmission, error	RDY	ARG	101	01000001	

MESSAGE TABLE (continued)

MNEMONIC	MESSAGE	CLASS	GROUP	CCC 210	DDDDDDDD 76543210	NOTES
ETO	end of transmission, OK	RDY	ARG	101	01000000	
GET	group execute trigger	CMD	ACG	100	00001000	
GTL	go to local	CMD	ACG	100	00000001	
IAA	illegal auto address	RDY	AAG	101	10011111	
IDY	identify			11x	xxxxxxx	
IEP	illegal extended primary	RDY	AAG	101	10111111	
IES	illegal extended secondary	RDY	AAG	101	11011111	
IFC	interface clear	CMD	UCG	100	10010000	
IMP	illegal multiple primary	RDY	AAG	101	11111111	
LAD	listen address 0-30	CMD	LAG	100	001aaaaa	1
LAG	listen address group	CMD		100	001xxxxx	
LLO	local lockout	CMD	UCG	100	00010001	
LPD	loop power down	CMD	UCG	100	10011011	
MLA	my listen address 0-30	CMD	LAG	100	001mmmmm	2
MSA	my secondary address 0-30	CMD	SAG	100	011mmmmm	2
MTA	my talk address 0-30	CMD	TAG	100	010mmmmm	2
NAA	next auto address 1-31	RDY	AAG	101	100nnnnn	3
NES	next extended secondary 1-31	RDY	AAG	101	110nnnnn	3
NMP	next multiple primary 1-31	RDY	AAG	101	111nnnnn	3
NOP	no operation	CMD	UCG	100	00010000	
NRD	not ready for data	RDY	ARG	101	01000010	

MESSAGE TABLE (continued)

MNEMONIC	MESSAGE	CLASS	GROUP	CCC 321	DDDDDDDD 87654321	NOTES
NRE	not remote enable	CMD	UCG	100	10010011	
NUL	null command	CMD	ACG	100	00000000	
OSA	other secondary address	CMD	SAG	100	011ttttt	4
OTA	other talk address	CMD	TAG	100	010ttttt	4
PPD	parallel poll disable	CMD	ACG	100	00000101	
PPE	parallel poll enable 0-15	CMD	ACG	100	1000sbbb	5
PPU	parallel poll unconfigure	CMD	UCG	100	00010101	
RDY	ready			101	xxxxxxxx	
REN	remote enable	CMD	UCG	100	10010010	
RFC	ready for command	RDY		101	00000000	
SAD	secondary address 0-30	CMD	SAG	100	011aaaaa	1
SAG	secondary address group	CMD		100	011xxxxx	
SAI	send accessory ID	RDY	ARG	101	01100011	
SDA	send data	RDY	ARG	101	01100000	
SDC	selected device clear	CMD	ACG	100	00000100	
SDI	send device ID	RDY	ARG	101	01100010	
SOT	start of transmission	RDY	ARG	101	01100xxx	
SRQ	service request	DOE or IDY		0x1 or 111	xxxxxxxx xxxxxxxx	
SST	send status	RDY	ARG	101	01100001	
TAD	talk address 0-30	CMD	TAG	100	010aaaaa	1
TAG	talk address group	CMD		100	010xxxxx	

MESSAGE TABLE (continued)

MNEMONIC	MESSAGE	CLASS	GROUP	CCC 321	DDDDDDDD 87654321	NOTES
TCT	take control	RDY	ARG	101	01100100	
UCG	universal command group	CMD		100	x001xxxx	
UNL	unlisten	CMD	LAG	100	00111111	
UNT	untalk	CMD	TAG	100	01011111	
ZES	zero extended secondary	RDY	AAG	101	11000000	

NOTES

1. 'aaaaa' represents a five bit binary device
   address in the range 0 to 30. The address
   may be either a primary listen or talk
   address or a secondary address, depending on
   the particular message. The least
   significant bit corresponds to bit D0.

2. 'mmmmm' represents the particular device
   address which matches the address assigned to
   this device.

3. 'nnnnn' represents the incremented device
   address which is transmitted by a device
   during auto address configuration. The range
   of this incremented value is 1 to 31. Note
   that 31 is not a valid address.

4. 'ttttt' represents a device address which
   does not match the address assigned to this
   device. The address may be either a primary
   talk address or a secondary address,
   depending on the particular message.

5. 's' represents the sense of a positive
    response to a parallel poll (IDY). If s is 0
    the device should do nothing for a positive
    response and should change its assigned bit
    to a 1 for a negative response. If s is a 1
    the device should change its assigned bit to
    a 1 for a positive response and should do
    nothing for a negative response. "Do
    nothing" means retransmit the frame
    unchanged.

6. 'bbb' is the binary bit number on which the
    device the device responds to parallel poll.
    000 means bit D0, 001 means bit D1, and so on
    to 111, which means respond on bit D7.

3. ELECTRICAL SPECIFICATIONS

3.1 General

This chapter presents the detailed electrical specifications for HP-IL. As discussed in chapter 1, the link between devices on the loop is a two-wire electrically balanced line. One of the conductors is designated the reference line and all voltage measurements are made with respect to the reference.

A device output must drive only one device input. This allows certain advantages over a bus type structure. Namely, the input may change the electrical signal to its own convenience without affecting any other device. Therefore, the specification thoroughly specifies a device output while putting only minimal constraints on a device input. Test procedures are specified for verifying the proper operation of input and output circuits to insure system compatibility.

Additionally, there may be instances where the designer requires certain additional capabilities from the link between devices that are not provided within this specification, such as longer distance or better noise immunity. As long as the waveforms and impedances at an HP-IL mechanically compatible output or input terminal are in compliance with the specification, the designer may choose any implementation desired. This might include level translation, data rate conversion, transmission media conversion, etc. For example, longer distances between devices could be achieved by permanently or uniquely attaching a cable to a special device output as long as the signal at the HP-IL compatible connectors are in compliance with the output specifications of section 3.3.

Any link with mechanically compatible HP-IL connectors, cables, etc. must also be functionally and electrically compatible with these specifications to prevent inadvertant connection to nonstandard links. Externally inaccessable links, as between multiple devices within a single package, may be implemented in any manner deemed appropriate.

These specifications are in general based upon an implementation using simple pulse transformers within the device. One transformer provides isolation and level conversion from the input terminals to the HP-IL receiver electronics and a second transformer performs a similar operation for the output terminals. While this specification is not intended to preclude implementations which do not use transformers, it is strongly recommended that transformers be used. It may in fact be quite difficult or even impossible to achieve these specifications with other technology.

logical ONE signal

logical ONE SYNC signal

logical ZERO signal

logical ZERO SYNC signal
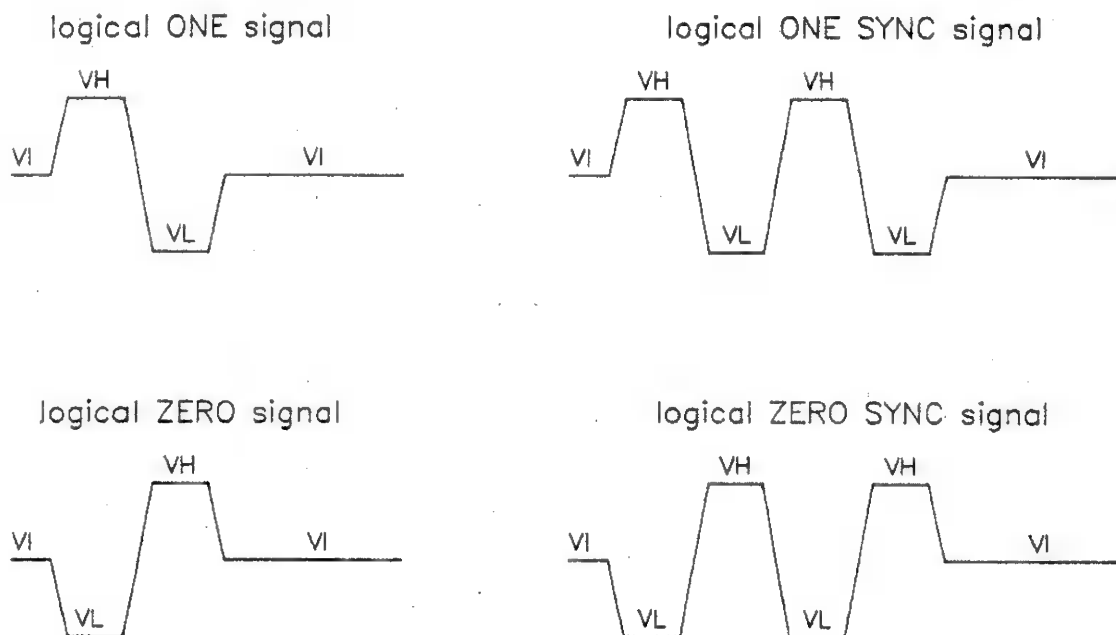
Figure 3-1.   HP-IL Electrical and Logical Relationships

3.2 Electrical and Logical Relationships

There are four distinct logical states defined by HP-IL as shown in figure 3-1. VH, VL, and VI are the voltage levels of the high, low, and idle times respectively. The detailed level and timing specifications are given in section 3.3.

The relationship between remote messages and the logical
states of the interface is as follows: The frame is divided into
a 'SYNC' code followed by ten (10) logical codes, each of which
is a 'Ø' or '1'. Encoded in the 'SYNC' code is the first bit of
the frame, C2, followed by C1, CØ, D7, D6, D5, D4, D3, D2, D1,
and DØ. A sample frame is shown in figure 3-2.

Interface Clear command frame
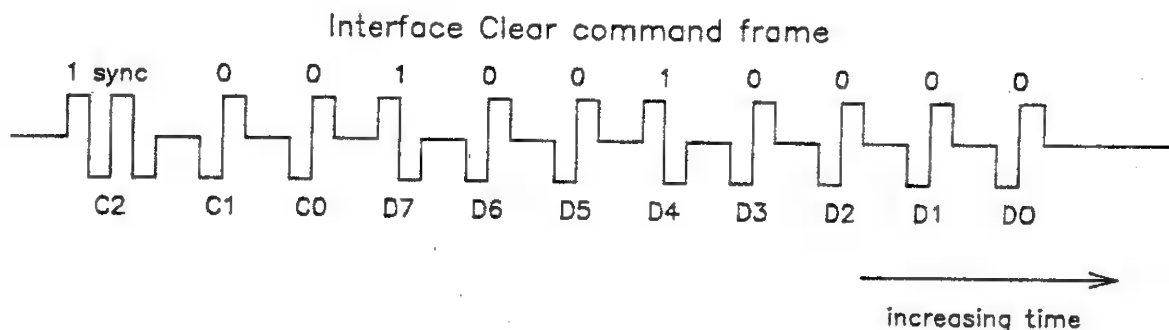
Figure 3-2. Sample HP-IL Frame

3.3 Output Specifications

The output is intended to drive a 100 ohm balanced
transmission line of any distance up to 100 meters in length.
The design is based on a matched source and unmatched load so
that transmission line reflections are absorbed at the source
(output terminals). The output is thoroughly specified in the
form of a Thevenin equivalent circuit so that the input circuit
design is given maximum information and freedom.

3.3.1 Open Circuit Output Voltage

The open circuit output voltage waveform shall remain
within the boundaries specified in figure 3-3. The measurement
shall be made at the device output terminals or at the end of an
HP-IL cable no more than one meter in length. If the HP-IL cable
is permanently or uniquely attached to the device, the
measurement shall be made at the cable output, even if its length
is greater than one meter (this allows for special drivers to
drive very long cables). It is recommended that the measurement
be made with a balanced differential oscilloscope such that the
impedance from each output terminal to earth ground is equal.

Note: While only a logical "1"
is shown here, all logical
bits must conform to the
same basic specifications.
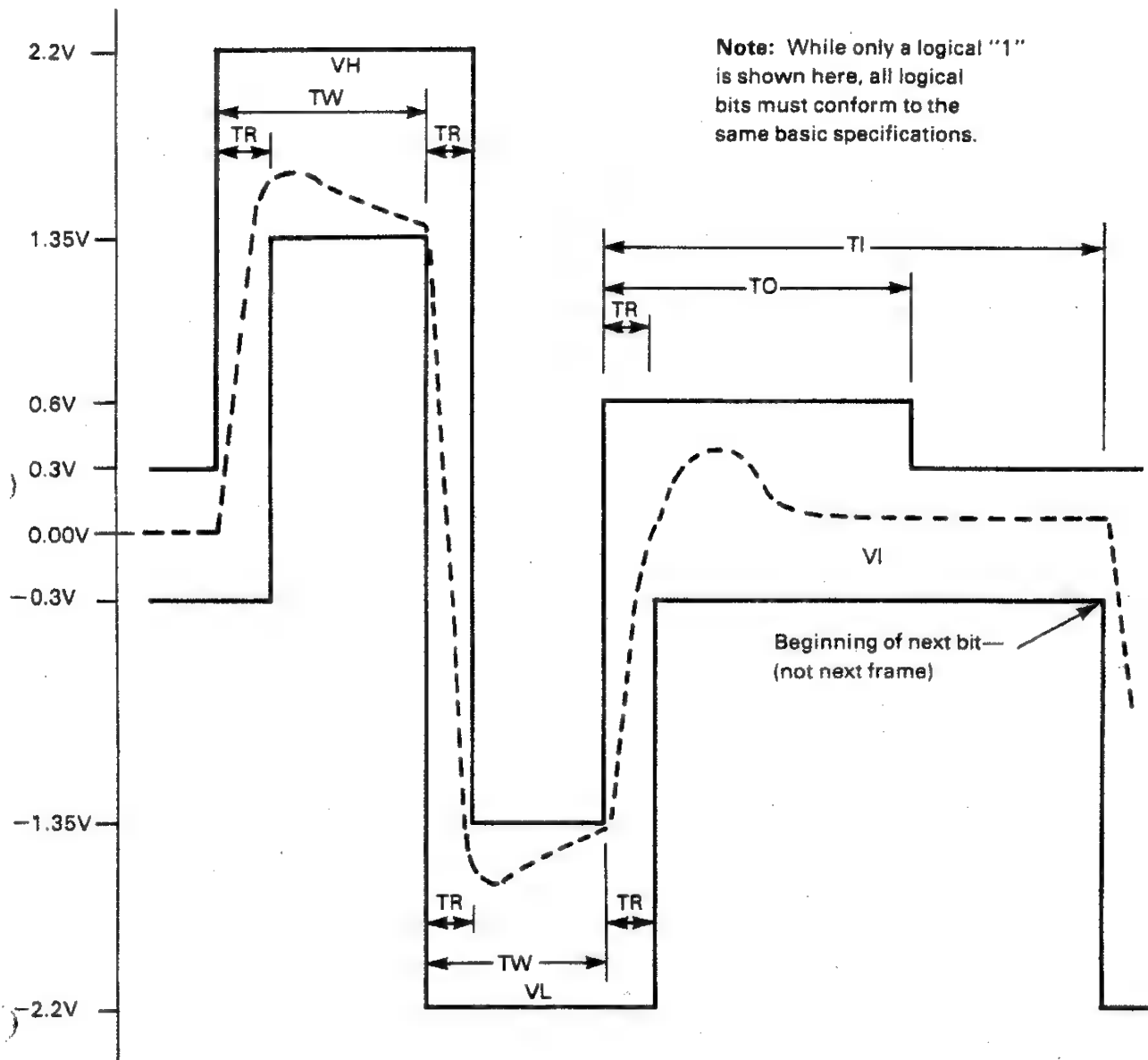
Parameter	Minimum	Maximum
TR	30nS	120nS (200 nS*)
TI	1.9μS	10μS
TO		1μS
TW	950nS	1500nS

*When tested with a 20 meter cable.

Figure 3-3.   Open Circuit Output Waveform

Note: A device which only barely meets the
waveform window of figure 3-3 is quite likely to
fail a waveform test at 20 meters unless the
output impedance can perfectly absorb all
transmission line reflections. This would
require output impedance to be identical to the
characteristic impedance of the cable. Refer to
section 3.3.2 for the test procedure of
reflection effects with a 20 meter cable.

The pulse width, TW, shall not vary from its mean value
by more than 10 percent within a single frame.

While the idle voltage, VI, extends to +-0.3 volts, the
DC voltage level shall approach 0 volts to prevent saturation of
an input circuit transformer. The large window for VI is
intended to allow for transient distortion caused by ringing,
reflections, etc. in the output circuits.

The idle voltage, VI, shall remain valid for at least 5
microseconds between frames.

3.3.2 Output Impedance

The output impedance shall remain within the boundaries
of magnitude and phase specified in figure 3-4 when the HP-IL
voltage levels are at the VH, VL, or VI levels. The measurement
shall be made between the two lines of the output, or at the end
of an HP-IL extension cable no more than one meter in length.
Conventional AC impedance meters are not capable of measuring
impedance during the VH and VL pulse times due to the narrow
pulse width and variable test signal amplitude. Therefore, it is
required that the output meet the tests outlined throughout the
rest of this section to insure system compatibility. It is also
the purpose of these tests to allow the maximum amount of design
flexibility by tolerating a significant amount of parasitic
reactance and/or non-linearity so that system size and cost may
be kept to a minimum. However, the allowance for design
flexibility requires more thorough testing than a more
restrictive specification.
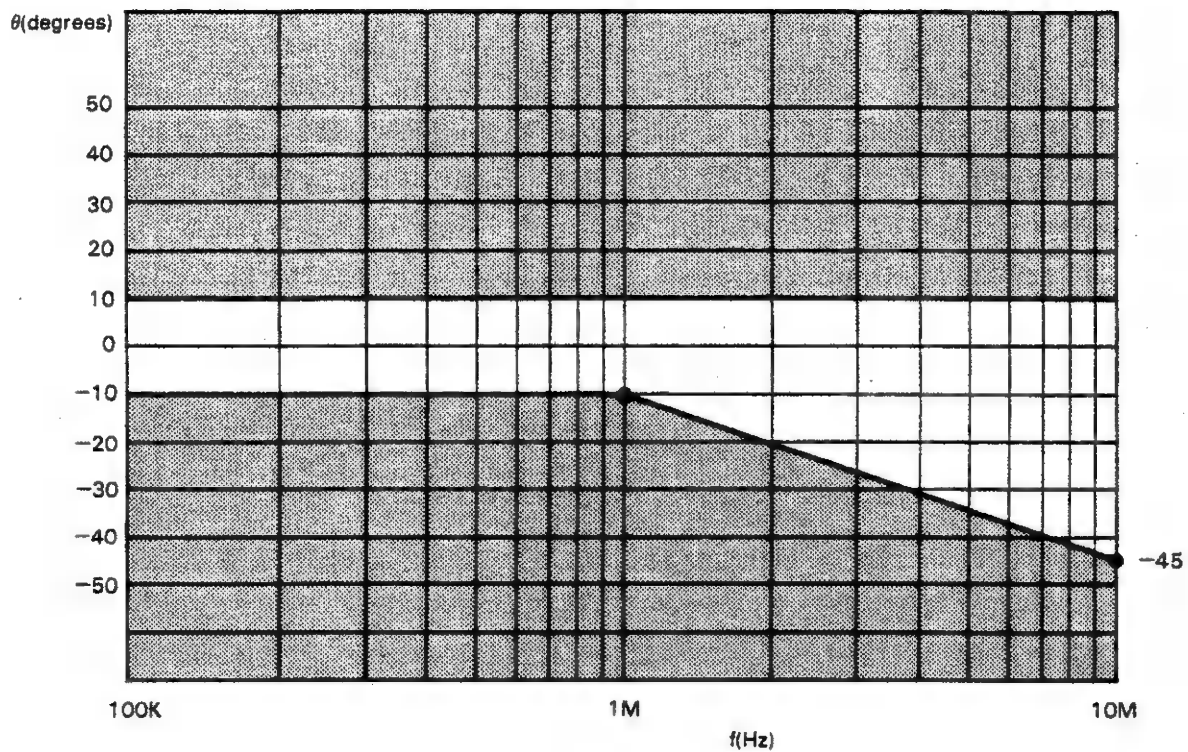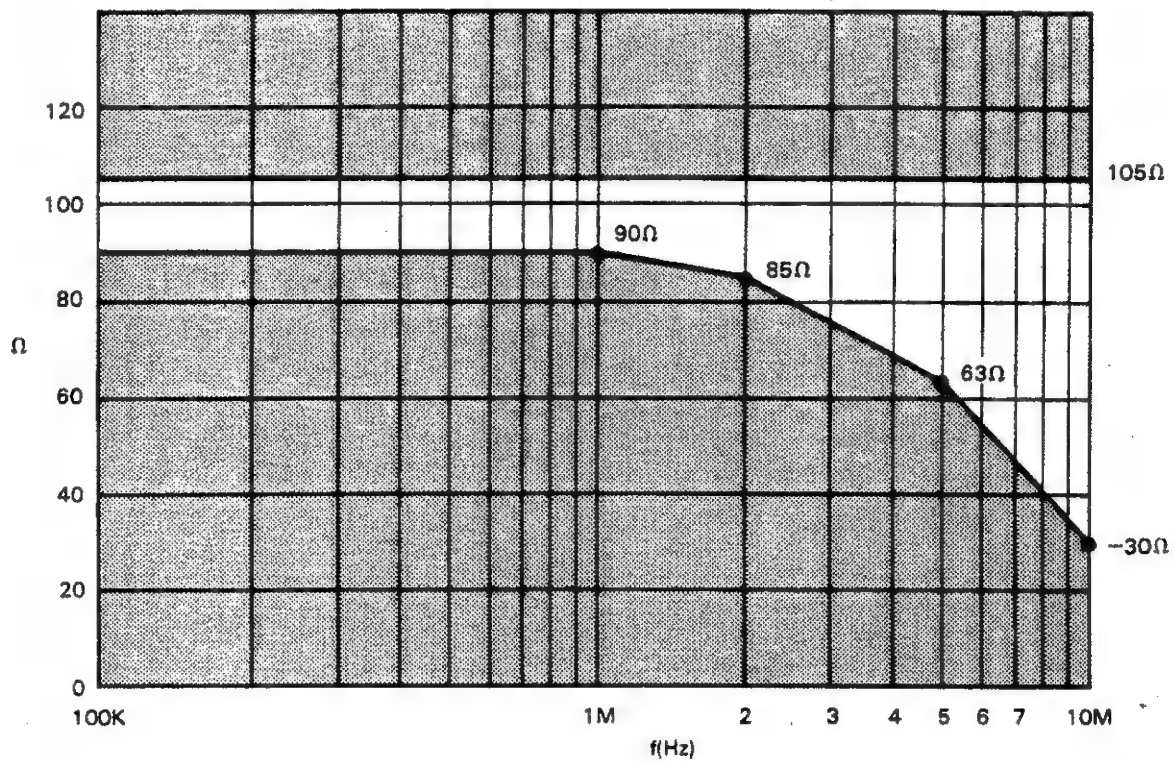
Figure 3-4.   Output Impedance Magnitude and Phase

Output Impedance - Pulse Attenuation and Droop Effects

At frequencies under 1 MHz, the output impedance is intended to put minimum and maximum boundaries on pulse attenuation caused by resistive loading at the input terminals of an HP-IL device, and also to put a limit on pulse droop caused by shunt inductive loading (due to transformer) at the input terminals. The actual amount of attenuation or droop is dependent on the input designer's choice of input resistance and inductance. Figure 3-5 outlines the test procedure to verify compliance. The test is done without an HP-IL input connected.

Spec: $90\Omega \leqslant |Z_{OUT}|_{midband} \leqslant 105\Omega$

Measurement limit:

$$\frac{V_{0.c}}{2.05} \leqslant V_{100} \leqslant \frac{V_{0.c}}{1.90}$$

V_{OC} and V_{100} shall be measured after rise time interval but before pulse droop is significant.

Observed Waveform
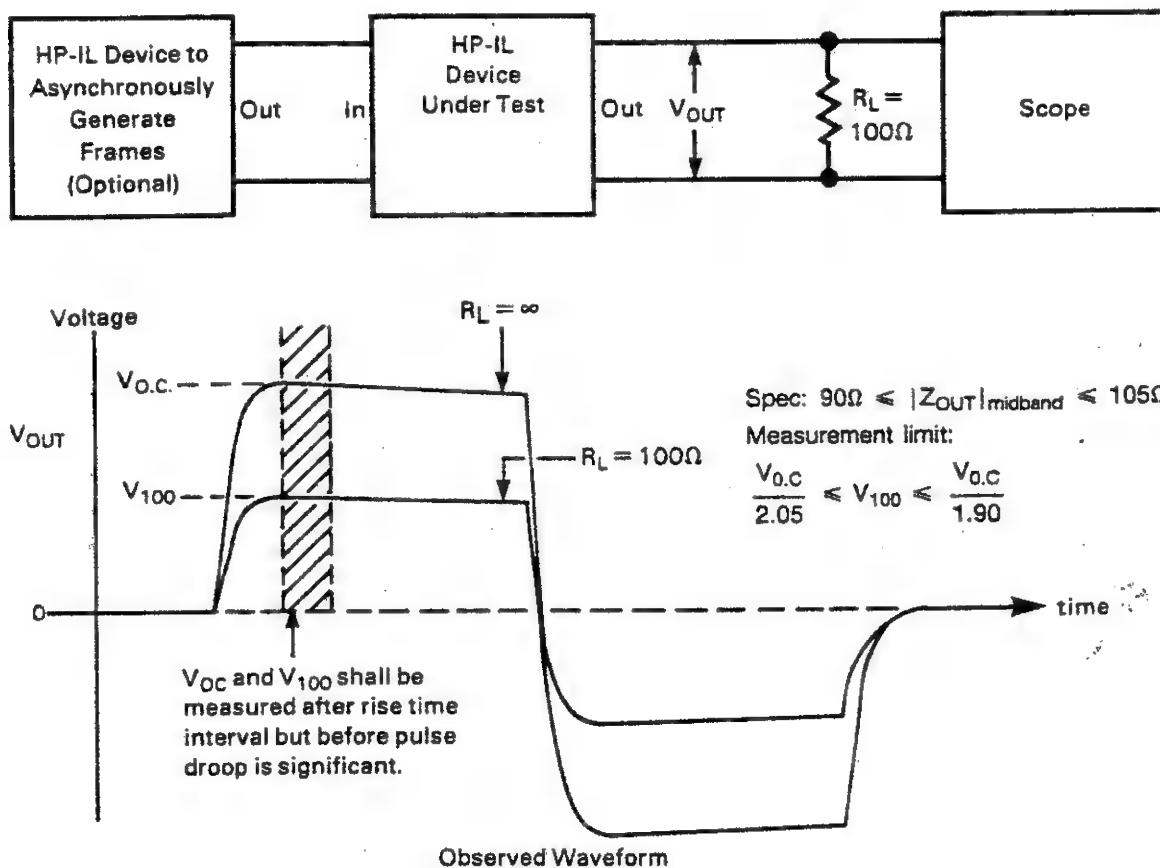
Figure 3-5.   Pulse Attenuation and Droop Test Procedure

The device under test must generate a frame(s) which allows voltage measurement of Vout. For convenient oscilloscope measurement, it is recommended that a single frame be repetitively generated. Since there is no input connected to the output under test, frames must be generated asynchronously by the device under test or by another HP-IL device if the device under test will repeat the frames. This may require some special device software since no handshake will occur.

Output Impedance - Resonance Effects

At short distances between devices, pulse distortion can occur in the form of pulse ringing or overshoot due to the presence of an underdamped RLC type circuit. Because of the probable presence of transformers in both the output and input circuits, the presence of parasitic inductance (L) and capacitance (C) is inevitable. The design objective is to keep the amount of ringing to an acceptable level. The output circuit is specified so that it does not contribute any significant parasitic inductance (by restricting positive phase shift to less than 10 degrees) at frequencies of importance to the input circuit (under 10 MHz). The output circuit is also specified to restrict the amount of parasitic capacitance (which could resonate with an input parasic inductance) by putting lower limits on impedance magnitude and phase shift in the 1 to 10 MHz range. Above 10 Mhz, the requirement that the waveform risetime be no faster than 30ns should provide adequate bandwidth limiting so that resonance may be ignored.

To test the above criteria, the following procedure is suggested: with the HP-IL output at the VI state (idle state), connect a vector impedance meter (e.g. HP 4275A or HP 4815A or equivalent) to the output terminals and measure magnitude and phase per figure 3-4 at frequencies from 100 KHz to 10 MHz at a test signal amplitude of 0.1 Vac. If it can be demonstrated that the output impedance is independent of test signal amplitude, a different amplitude may be used.
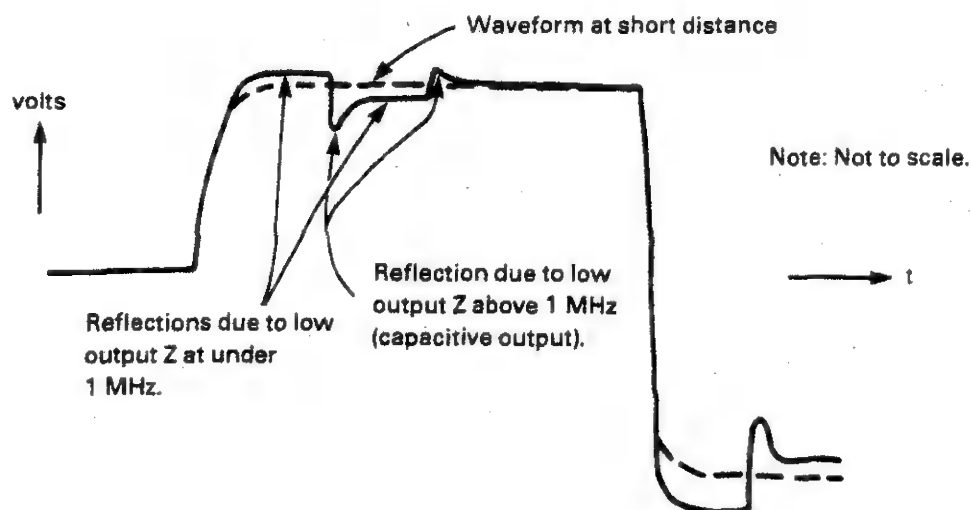
Figure 3-6. Output Waveform With Reflections

Output Impedance - Reflection Effects

At relatively long distances between devices (more than
10 meters), the output impedance is intended to sufficiently
match the cable characteristic impedance (100 ohms) so that
transmission line reflections are adequately absorbed. For
example, an output impedance which barely meets the lower limit
of impedance magnitude (figure 3-4) might generate a waveform at
the end of a long cable as shown in figure 3-6.

Non-linearities in the output impedance might also cause
reflections which a small signal impedance measurement might
overlook. Therefore, it is necessary that the output voltage
waveform remain within the boundaries of figure 3-3 when tested
according to figure 3-7. The device under test must generate
frames which allow measurement of the output voltage. Another
device may be used for this purpose as shown in figure 3-5. The
20 meter cable is long enough to generate distinct reflections,
yet short enough to prevent cable losses from filtering spike
type reflections caused by a reactive source impedance. The
capacitive load is representative of filtering required of the
input circuits of high frequency spike type distortion.
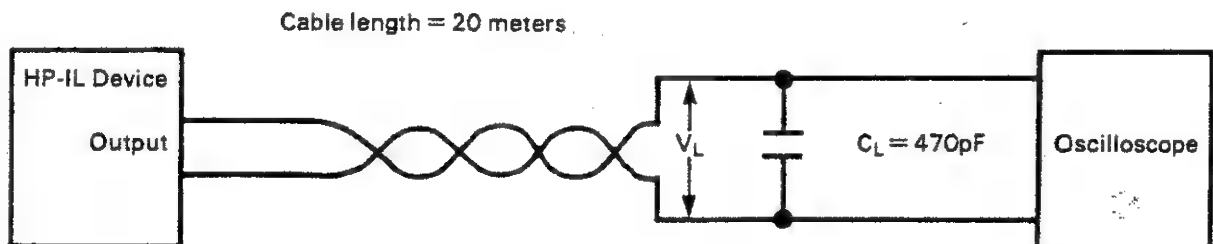
Figure 3-7. Output Impedance Test - Reflection Effects

3.3.3 Common Mode Output Voltage

Since the loop is not perfectly isolated from earth,
there is the possibility that during normal operation a device
could source common mode signals with respect to earth. The
input circuitry is required to reject a certain amount of common
mode signal (refer to section 3.7), but above that amount errors
can occur. In addition, common mode signals can cause currents
to flow through the capacitance present at the loop inputs if the
signal is changing with time. Therefore, the amount of common
mode signal that the output circuitry can source is restricted to
a level that all input circuits can reject. In addition, the
rate of change of the common mode signal is limited.

It should be emphasized that these problems can occur during normal operations of the device. There will always be abnormal conditions that could cause a device's output signal to exceed the allowable common mode specifications (e.g. electrostatic discharge).

Under normal operating conditions, the common mode signal sourced by the device under test should meet the restrictions listed in figure 3.8.

The diagram shows:

D.U.T. with Out, two 50Ω resistors, capacitor C to Earth, output V_{CM}.

$C \leqslant 25pF$:

$V_{CM} < 500$ volts peak

$\dfrac{dV_{CM}}{dt} < 100V/\mu s$

$C = 100pF$:

$V_{CM} < 300$ volts peak

$\dfrac{dV_{CM}}{dt} < 100V/\mu s$
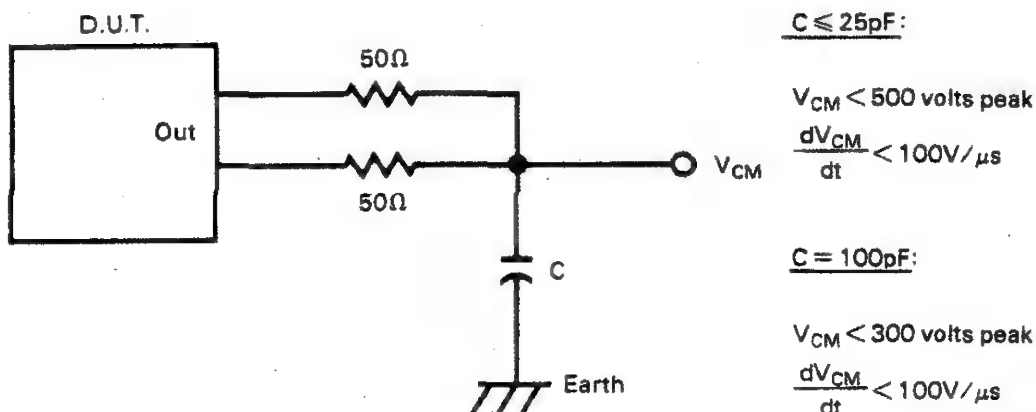
Figure 3-8.   Common Mode Output Voltage Test

3.4 Input Specifications

Because each HP-IL output drives only one input, the circuit designer is given as much freedom as possible to detect the specified output waveform using whatever attenuation or waveshaping techniques desired. However, this requires the designer to thoroughly understand how the voltage waveform is modified by different combinations of output, input, and interface cable circuts.

3.4.1 Input Impedance

To limit the amount of power dissipation of the sourcing device, the input resistance shall be at least 100 ohms when measured at 100 KHz at a voltage level of up to 0.85 volts AC (2.4 volts peak to peak). At voltage levels greater than 1.2 volts or less than -1.2 volts, the input impedance is not specified. This allows an input to clamp the voltage, if desired, for waveshaping, electrostatic discharge protection, etc. The chosen level is above the maximum voltage generated by the output circuit if the input is a totally linear 100 ohms.

The input reactance is not specified. However, the input designer must use care to insure that any input reactances, parasitic or otherwise, do not create any waveform distortion which might adversely affect signal detection reliability.

3.4.2 Input Hysteresis

To provide a minimum level of immunity to externally induced noise or internally generated distortion, it is strongly recommended that the input circuits use a schmitt trigger type input with hysteresis.

3.4.3 Input Test Circuits

Because the input circuit thresholds or timing requirements are not specified, it is necessary to provide a means of testing the input circuits for proper operation in all possible HP-IL system configurations. To achieve this, test circuits are specified which are representative of several worst case configurations. While these circuits do not cover all posible worst case configurations, they represent a minimum level of testing to insure compliance.

Table 3-1 lists the characteristics of 5 test configurations to be used to test the input of the device. With the exception of pulse amplitude and output impedance phase, the output parameters are taken directly from the worst case output specifications of section 3.3. A modest amount of noise margin has been included for pulse amplitude and impedance phase to allow for test measurement error and to improve system reliability.

Table 3-1.　Input Test Configurations

	Test Circuit	Open Circuit Output Waveform (see figure 3-3)					Output Impedance			Cable Length
		Pulse Amplitude (VH,VL)	Idle Amplitude (VI)	Rise Time (TR)	Pulse Width (TW)	Idle Time (TI)	Magnitude (|Z|)	Phase (θ)	Test Frequency	
1	1	1.30v	—	120ns	950ns	1900ns	105 ohm	0°	1 Mhz	0 m
2	1	1.30v	—	120ns	950ns	1900ns	105 ohm	0°	1 Mhz	100 m
3	2	2.30v	max.*	< 50ns	1500ns	—	90 ohm 64(−3dB)	0° −450°	100 KHz 3 MHz	0 m
4	2	2.30v	max.*	< 50ns	1500ns	—	90 ohm 64(−3dB)	0° −450°	100 KHz 3 MHz	20 m
5	2	2.30v	max.*	< 50ns	1500ns	—	90 ohm 64(−3dB)	0° −450°	100 KHz 3 MHz	100 m

* Recommended only — may be difficult to generate

　　　　The two test circuits required to generate the necessary worst case waveforms may be obtained by modifying an HP-IL device's output circuits to meet the requirements of table 3-1. While the means to achieve this are very device dependent, an example is shown in figure 3-9.

　　　　The first test circuit is primarily intended to guarantee the detection of the pulse levels VH and VL, the idle level VI, and to guarantee that the VH to VL transition is not detected as a VI state between bits. The second test circuit is primarily intended to guarantee that extra pulse levels, VH and VL, are not detected following a bit. This might cause a normal bit to look like a SYNC bit. This could be caused by ringing, reflections, pulse droop, etc. The type of distortion is quite different at various cable lengths and thus must be tested at three distances.
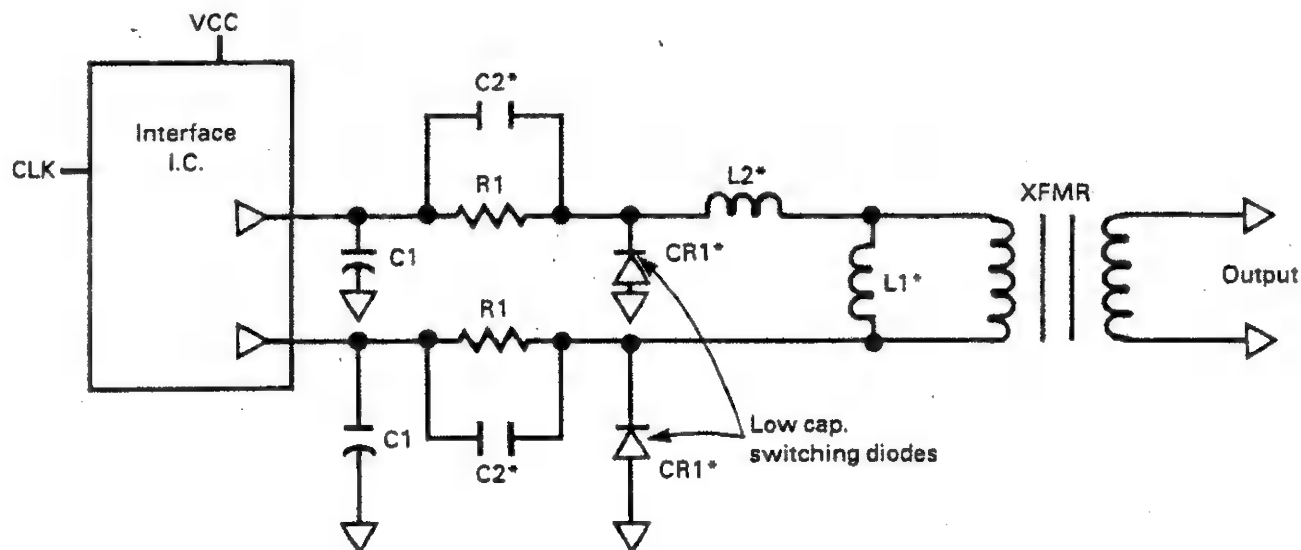
* Components added to normal circuit to create test circuit

Notes:
1. VCC should be adjusted to set pulse amplitude (VH,VL).
2. CLK should be adjusted for pulse width or idle time (TW,TI).
3. C1 should be adjusted to set rise time (TR).
4. Switching diodes can create a non-linear source impedance
    which can force idle amplitude to a maximum level because of
    transmission line reflections. Diode capacitance must be low
    to allow independent control of rise time.
5. R1 should be adjusted to set output impedance at 100KHz.
6. C2 should be adjusted to set frequency at which output
    impedance is down 3dB.
7. L1 may be added to adjust pulse droop if desired.
8. L2 may be added to increase pulse overshoot if desired.

Figure 3-9:   Input Test Circuit Example

3.4.4 Input Filtering

There are several ways in which a waveform at a device input may be distorted. Externally generated electromagnetic interference (EMI) may affect the signal received by the device. The reactive portion of a device's output impedance will cause some waveform distortion (ringing, reflections, etc) generally in the 2 to 10 MHz range. Also, an input circuit using transformer isolation is likely to generate some distortion in this frequency range due to parasitic reactances.

Decode as a logical ONE   Decode as a logical ZERO
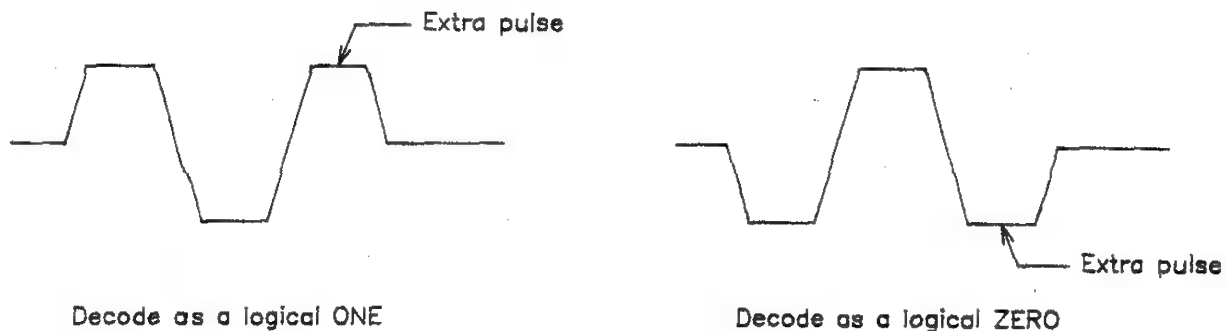
Figure 3-10:   Extra Pulse Filtering

To enhance reliable signal detection, it is strongly recommended that some form of filtering be used to reject this distortion. In addition, the input shall ignore extra pulses as shown in figure 3-10.

3.5 Interface Cable Specifications

The interface cable between any two devices is a two wire balanced type cable of no more than 100 meters. A cable which is permanently or uniquely connected to a device output shall not be considered part of this 100 meter constraint. Thus, the distance between two devices may be lengthened arbitrarily if the device has a cable permanently or uniquely attached (i.e. dedicated cable) and meets the input/output specifications as measured at the end of the unique cable.

Throughout this section, references to cable specifications are to be interpreted as the entire link between one device's output terminals and another device's input terminals. If 2 or more cables are connected together between devices, the total length of all cables connected must meet the specifications listed in this section.

3.5.1 Cable Type

For distances less than 10 meters, there is no restriction on the type of two wire cable used. Therefore, a very low cost cable may be used as long it meets the specifications. Longer distances, however, require a shielded twisted pair. The shield prevents the characteristic impedance from being affected by environment or application (coiling, conduit installation, etc), and also helps balance EMI noise pickup onto each conductor to limit differential mode noise. The conductor twisting also helps balance noise pickup. The shield may be left unconnected or may be connected to earth ground provided the connection is made only at the input terminals of a device. If the shields of more than one interface cable are connected together, then grounding may only occur at one input terminal. The shield shall not be connected to device common if it is not also connected to earth ground.

3.5.2 Characteristic Impedance (Z0)

Characteristic impedance control is necessary to limit transmission line reflections. Reflections are less significant at short distances than they are at long distances, but they cannot be ignored. Z0 requirements:

$$Z0 = 100 \text{ ohms} \pm 20\% \quad (\text{cable length} <10m)$$

$$Z0 = 100 \text{ ohms} \pm 10\% \quad (\text{cable length} >10m)$$

The characteristic impedance is measured using Time Domain Reflectometry techniques or equivalent. A suggested test procedure is given in figure 3-11.

Note: All lead lengths must be kept to absolute minimum.

Slowly rising voltage results from distributed reflections caused by cable loss. V_1 should be measured at leading edge.
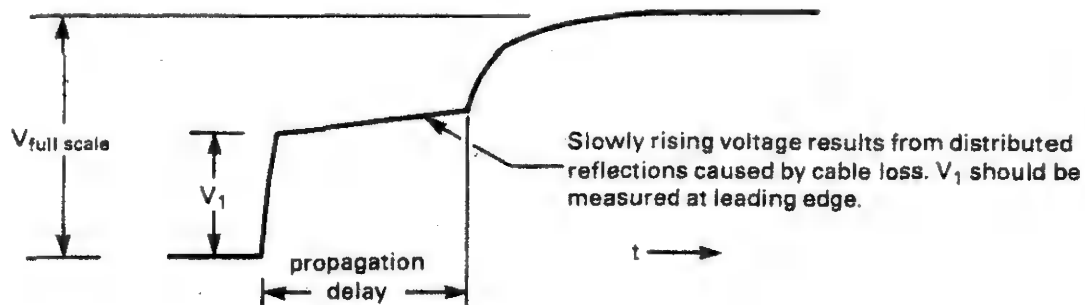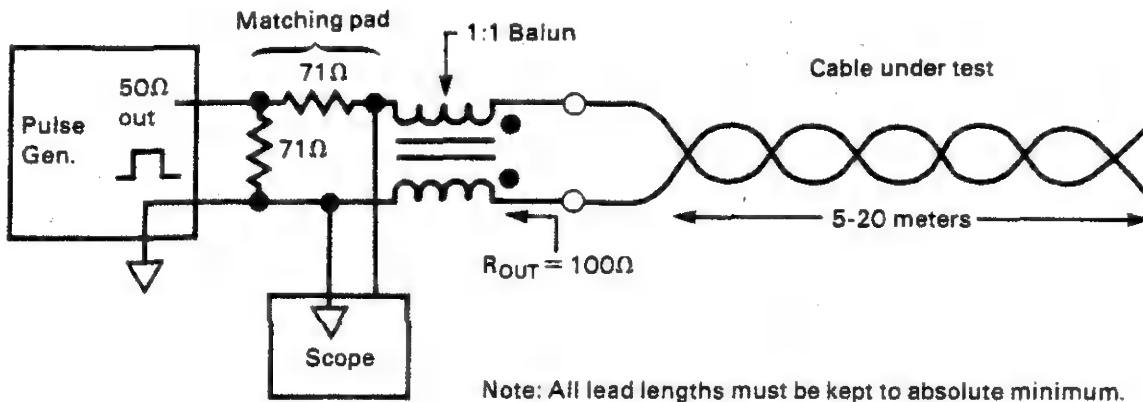
Procedure:

1. Adjust pulse width to be much greater than propagation delay.
2. With cable removed, adjust pulse amplitude for full scale deflection on scope.
3. Check test circuit output impedance by replacing cable with a 100 ohm resistor and verifying 1/2 full scale deflection.
4. Connect cable under test and measure V1.

$$Z0 = \frac{(V1 * Rout)}{(Vfull \ scale - V1)}$$

Figure 3-11.   Characteristic Impedance Test Procedure

3-16

3.5.3   Cable Rise Time

The 10% to 90% cable rise time must not be more than 200 nanoseconds when measured per figure 3-12 or equivalent. The cable(s) under test should be configured for the total length to be tested because rise times increase exponentially with distance (not linearly).
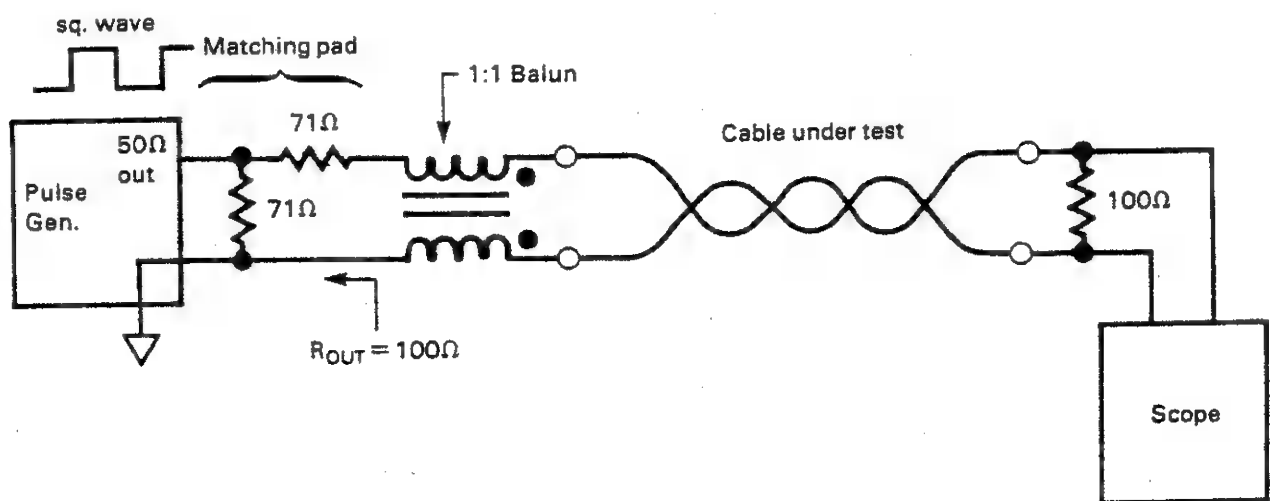
Procedure:

1. Measurement shall be made at the 100 ohm termination with a a balanced differential oscilloscope.
2. Make pulse width long enough so that all reflections and transients have time to settle to a steady state value.
3. Adjust the steady state voltage level for full scale deflection on scope.
4. Measure 10-90% rise time relative to the steady state values.

Figure 3-12.   Cable Rise Time Test

3.5.4 Cable Loss

Cable losses are caused by series and shunt losses and are frequency dependent, being worst at high frequencies. The main result is to slow cable rise time (along with dispersion effects). Therefore, the rise time specification is sufficient to adequately specify high frequency losses. However, the rise time is specified relative to a steady state or DC level. For an open circuit termination, the steady state cable output will equal the source voltage. However, for a finite termination resistance, the cable DC series resistance will cause a loss for which the designer of an HP-IL input circuit must compensate (refer to figure 3.13).
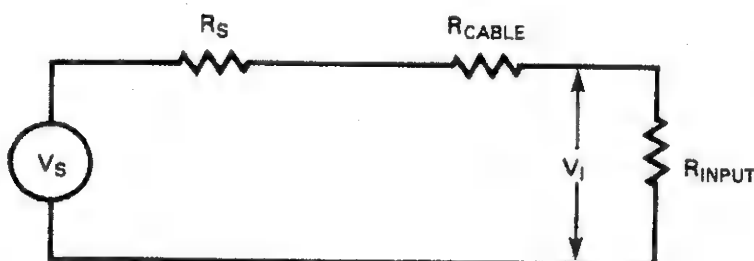
$$V_I = \frac{R_{INPUT} \times V_S}{R_{INPUT} + R_S + R_{CABLE}}$$

Figure 3-13.   DC Cable Losses

The total DC cable resistance between two devices must not be more than 7 ohms on each conductor. For a 100 ohm output and input impedance, this would result in a 46.7% steady state attenuation, compared to 50% attenuation for a lossless line.

3.6 Isolation Requirements

Both signal lines of the input and output terminals must be isolated from earth (chassis or safety) ground and device common (for devices without earth ground) as follows:

Capacitance < 100 pf

Resistance > 10 MegOhms (40 deg C, 80% rel. humidity)

Voltage Breakdown > 500 V dc

3.7 Electromagnetic Compatibility (EMC)

The EMI emissions of the interface shall be in compliance with VDE and FCC class B radiated and conducted interference requirements for cable lengths under two meters. The interface shall comply with class A requirements for cable lengths greater than two meters.

Susceptibility to EMI in HP-IL devices must be limited as much as possible because the HP-IL system may be used in environments which are subjected to many sources of EMI. One of the most severe sources is from transients on power lines in close proximity to the HP-IL cable. For more information on this subject, refer to General Electric's Transient Voltage Suppression Manual*.

It is assumed that HP-IL devices will perform error checking and that many devices will perform error handling and recovery such that only a small number of errors would require additional action from a user. Therefore, the specification allows for a non-zero error rate that insures reliable operation in most normal environments.

Because of the balanced nature of the interface cable, most of the EMI picked up by the cable will be in the form of a common mode signal. Therefore, this section will primarily test for the common mode rejection ability of an HP-IL device. While not explicitly specified, this will require the devices to maintain a reasonable amount of balancing, isolating, and/or shielding within the input and output circuits. Refer to figure 3.14 for test procedure.

* Transient Voltage Suppression, 3rd ed. (New York: General Electric Company, 1982)

Figure: Line Transient Test setup showing Transient Generator connected via Unshielded power cord (2m, 1cm spacing) to HP-IL cable, with OUT D.U.T. IN and IN D.U.T. OUT positioned 0.5m below, connected to Controller.
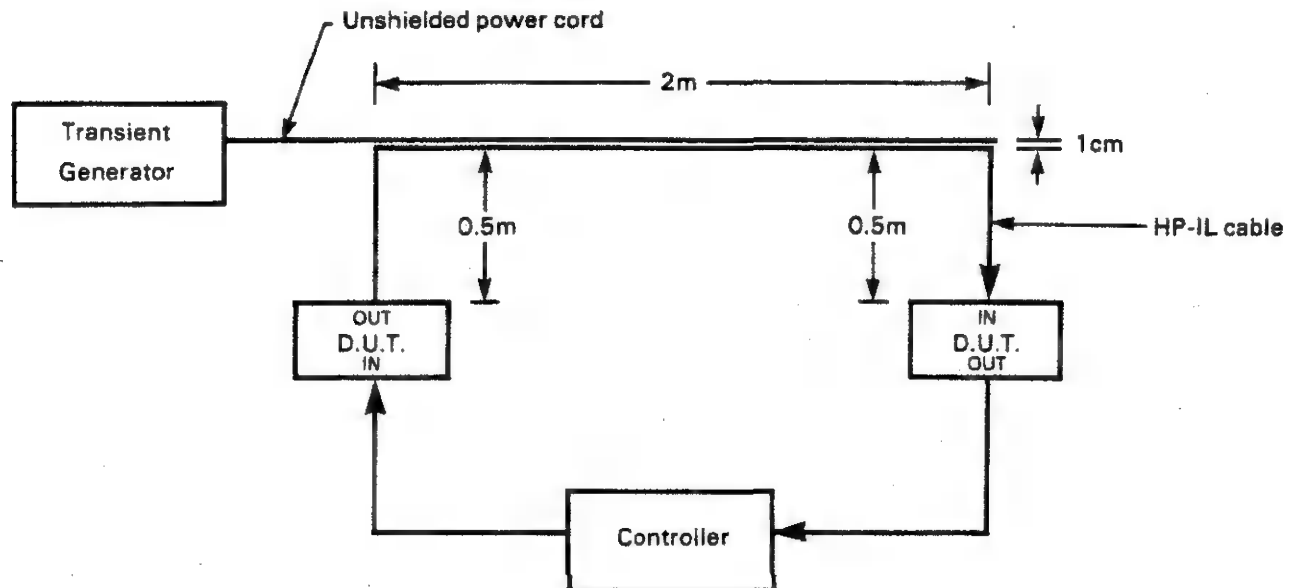
Procedure:

1. Transient Generators:
   Schaffner 222 or equivalent (5ns rise time)
   Velonex 360 or equivalent (1.25MHz damped sine wave)
2. Identical units of the Device Under Test (D.U.T.) should be located as shown. If only one unit can be tested, it must be tested in both positions shown to test both the input and output circuits, and the controller's input and output circuits must comply with this section.
3. The D.U.T.s shall be tested on and off ground planes, and with AC power source connected if applicable.
4. The controller shall generate an average of more than 1000 frames per second. Error checking must be performed and the number of errors must be counted.
5. An oscilloscope should be used to verify the actual frame rate prior to the test. The oscilloscope should not be connected to the HP-IL interface during testing.
6. Both common and differential mode line transients should be applied to the power cord.
7. Because the line transients and the HP-IL frames are not synchronized, the test shall be run for more than five minutes and the total number of errors averaged for that interval.
8. Line transient repetition rate shall be set to maximum: 30Hz on Shaffner 222, 100Hz on Velonex 360.

Figure 3-14.   Line Transient Test

Table 3-2:　EMI Test Limits

Transient Generator	Amplitude	Normalized Error Rate* (max)
Schaffner 222 (rise time set to 5ns)	500V	2.0 E -7
	1000V	2.0 E -6
Velonex 360 (IEEE 472)	500V	5.0 E -7

* Normalized error rate:

$$\frac{\dfrac{\text{Total number of errors}}{\text{test time (seconds)}}}{\text{Frame rate (Hz) * transient rate (Hz)}}$$

For example, a 1KHz frame rate, a 30Hz transient rate on the Schaffner 222, and a 1000V test amplitude, the maximum allowable number of errors in a 5 minute test would be 18.
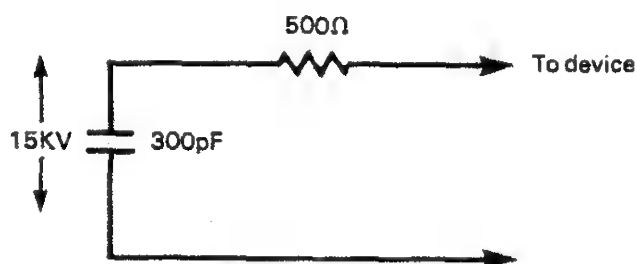
Figure 3-15.　ESD Equivalent Circuit

3.8 Electrostatic Discharge (ESD)

HP-IL devices shall not experience any permanent failures as the result of a 15KV potential applied to any point on the interface, including the cable conductors. The equivalent circuit is shown in figure 3-15. This test represents a situation where a statically charged person may touch the interface, or may touch a device on the interface which couples the resulting discharge onto the interface cable.

It is strongly recommended that no temporary errors (data loss, resets, etc.) occur when this test is performed.

4. MECHANICAL SPECIFICATIONS

As mentioned previously, due to the special requirements
of HP-IL a special connector set is used. Devices may either use
a panel connector or cables affixed directly to the device
through strain reliefs (figure 4-1). It is strongly recommended
that devices use panel connectors in cases where a choice is
available. Cable connectors (figure 4-2) are designed to be non-
invertible and non-reversible and have a positive detent both in
panel connectors and in each other for running cable "splices".
In short, they are intended to be as foolproof as possible.

The connector contacts were chosen for high reliability
and long life. The connector body is of molded polycarbonate
while the integral strain relief is of PVC.

For applications where the distance from one device to
the next is 10 meters or less, relatively inexpensive "zip" cord
cable may be used. The wires are 24 AWG stranded (26 X 38 AWG)
individually tinned copper. The wires are spaced 0.060 inches
center-to-center with a PVC jacket, 0.065 X 0.130 inches. A
polarity rib must be visible on the jacket.

For applications requiring longer distances up to 100
meters, cable which satisfies the more stringent electrical
specifications given in the previous chapter must be used, that
is, shielded twisted-pair cable whose characteristic impedance is
100 Ohms plus or minus 10%. Refer to chapter 3 for electrical
characteristics of the cable.

Mtg. Holes
1.330

PC Tails or
Lead Term.

.625

.080

Ref. Line

OUT

.400

IN

1.550

Panel Cut Out   .187R

1.160

Rib

.440

Panel Cut Out

.210

.300

.300
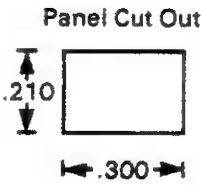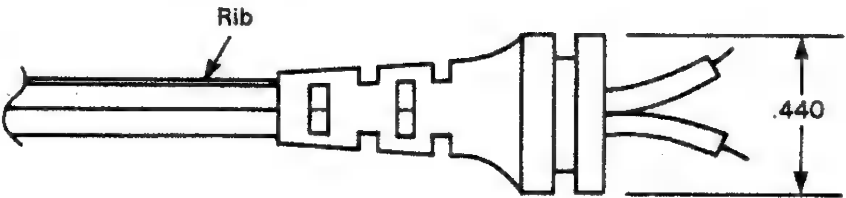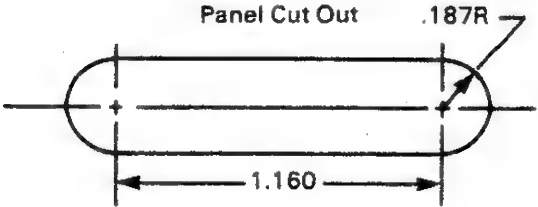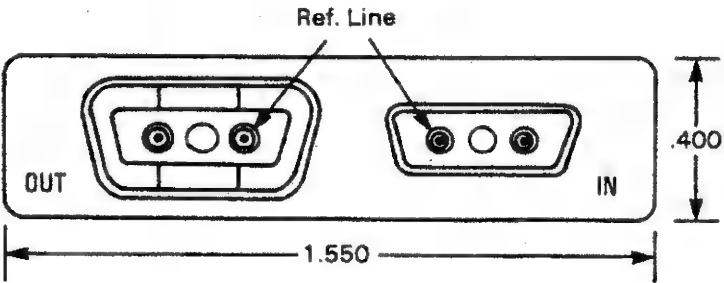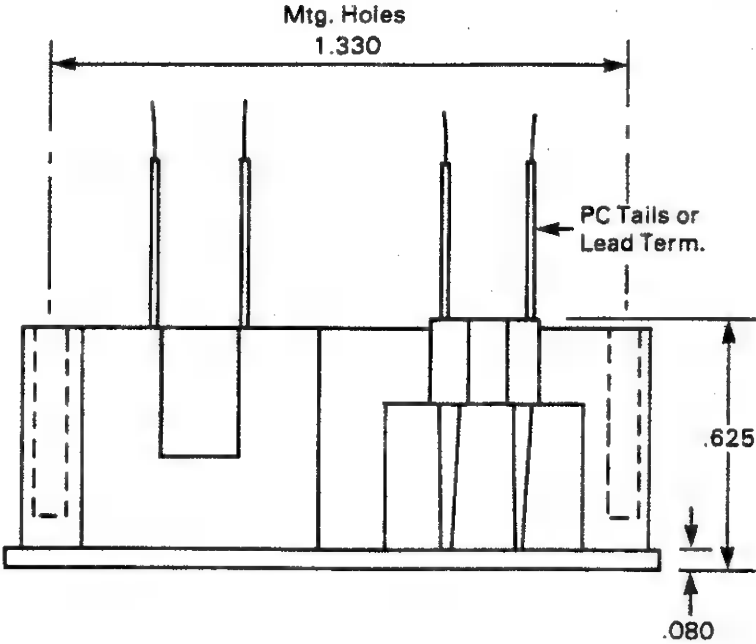
Figure 4-1.   Example Device Connectors

.63

.38

Ref
Line

.63

.38

Ref
Line

Rib
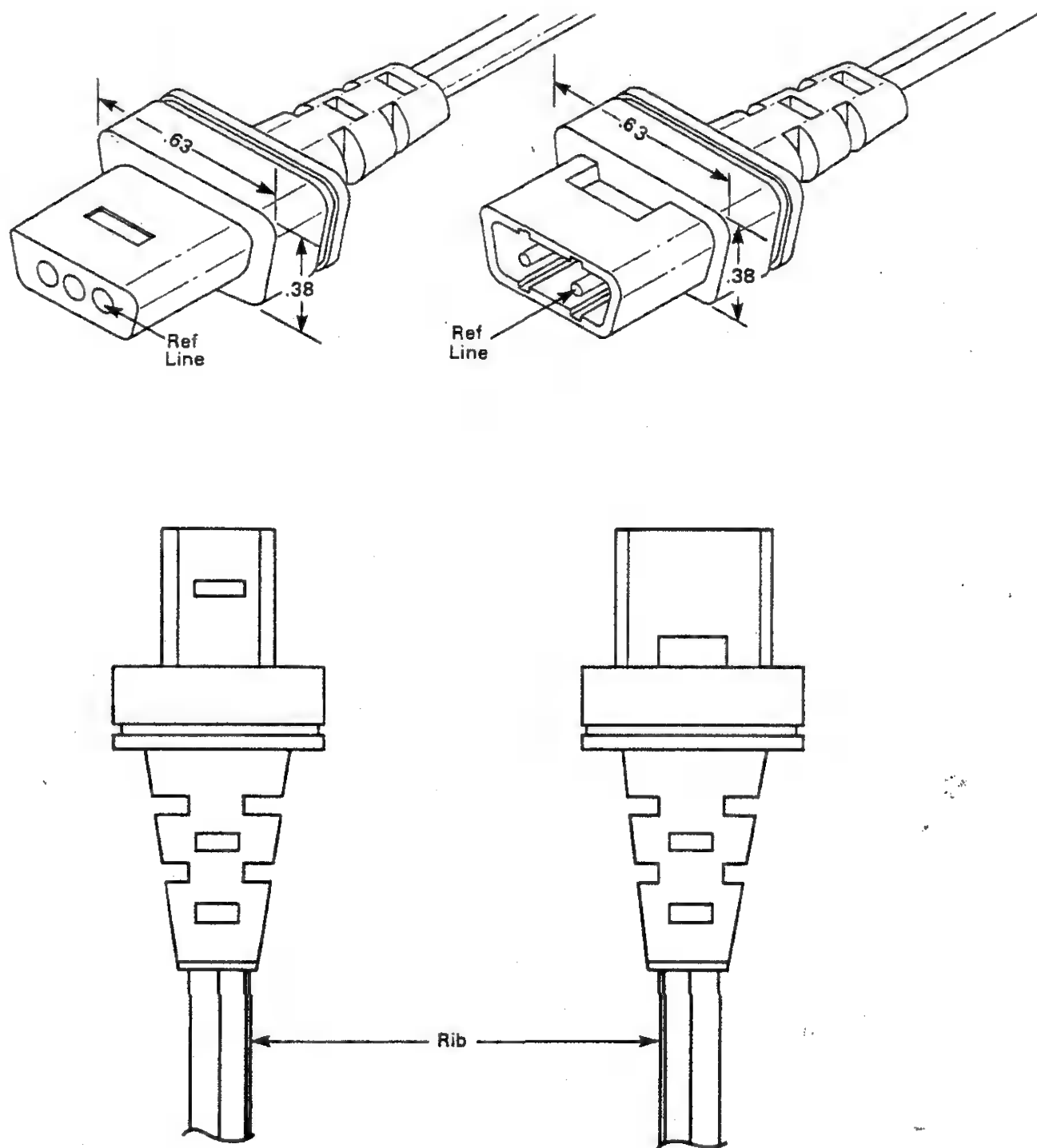
Figure 4-2.   Cable Connectors

5. SYSTEM GUIDELINES

5.1 System Compatibility

This interface sytem offers a wide range of capability from which to choose the appropriate interface functions to fit different applications. Within most interface functions a number of options are available. In addition, the designer has freedom to select all the device dependent capabilities contained within the device functions.

It is the responsibility of the designer to define the complete capability of a device (interface function choices and related device dependent interactions) so that the end user of the device can efficiently interface and program the device for appropriate system applications.

Also, devices designed to this interface system may have a wide range of capability relative to their ability to communicate over the interface. This document does not cover the operational characteristics of devices, only the functional, electrical, and mechanical capabilities of the interface system.

The burden of responsibility for system compatibility at the operational level is on the user. The user must be familiar with all device characteristics interacting with the interface system (for example, device dependent program codes, output data format and codes, etc.).

In particular, compatibility will be greatly enhanced if the ASCII code is used by all devices for device dependent messages whenever possible. In addition, it is recommended that device dependent commands be avoided unless absolutely necessary. Simple ASCII programming codes or ASCII escape sequences are preferable for the implementation of simple "friendly" controllers.

5.2 System Configuration

If all devices on the loop use only one byte addresses, the maximum number of devices in an HP-IL system is 31. If all devices use extended addressing, the maximum number is 961.

In general, an interface system shall contain one or more devices containing at least one T function, one L function, and one C function.

If a T function includes the use of the local message ton and at least one L function includes the use of the local message lon, a system may be operated without a C function while the ton message is true in one and only one T function and the lon message is true in one or more of the L functions. The ton and lon messages are normally provided by local switches. This permits rudimentary, manual system operation in which one talker sends data continuously to one or more listeners.

All system configurations containing more than one controller must satisfy the following conditions: (1) There shall not be more than one C function in a system which is sending the scl local message true at any time. (2) Every controller in the system shall be able to pass and receive control of the interface.

Because of the serial nature of HP-IL, all devices must be powered on in order for the system to function. If desired, the designer may implement the interface circuitry such that when the device is powered off, power remains on for the interface circuitry to act as a simple repeater.

5.3 Address Assignment

Normally, a device will be assigned a single talk and a single listen address to perform the essential tasks. It may be useful to design a device with multiple talk (or listen) addresses to facilitate system requirements. A device could be assigned two talk addresses, for example, one to output raw data, the other to output processed data. If this is necessary, it is recommended that two-byte addressing be used so that limited one byte addresses will still be available.

When devices are powered on they may either respond to no address at all or they may respond to a default or switch selectable address. Responding to no address reduces the chances of two devices having the same address (which could happen if a device is reset during normal operations). When the AAU (auto address unconfigure) command is received, devices must respond to a preset or switch selectable address. Note that some devices require the AAU message to be sent before default addressing may be used.

It is important to note that the auto addressing messages have no effect on the T or L interface functions. All messages that affect the T and L functions are defined in chapter 2 and do not include any auto address messages. Therefore, if a device is in an addressed state when the AAU message arrives, it will remain in that state and respond accordingly to messages which follow. Normally, the messages which follow will assign new addresses. The act of assigning addresses to particular devices is distinct and separate from the act of addressing a particular device to talk or to listen. Refer to the appropriate sections of chapter 2 for more information.

A device that contains the T function may be assigned any value for the least significant five bits of its MTA (my talk address) message code other than 11111. This code, defined as UNT, is provided for the controller to return all devices to the talker idle state. Two or more T functions shall not be assigned the same value for these bits unless different secondary addresses are assigned. A device that contains both a T and an L function should have the same value for the least significant five bits of both the MTA and MLA (my listen address) messages.

A device that contains an L function may be assigned any value for the least significant five bits of its MLA message code other than 11111. This code, defined as UNL, is provided for the controller to return all devices to the listener idle state.

A device that contains extended talker or listener functions may be assigned any value for the least significant five bits of its MSA (my secondary address) message code other than 11111. Two or more devices with the same value of their MTA codes may not have the same value for MSA codes. If a device has both extended talker and listener functions, the lower five bits of the MTA and MLA codes should be equal and both functions should use the same MSA code as well.

In general, and particularly when auto addressing or auto extended addressing is used, the T and L function within a specific device will be assigned and will use the same address. Auto multiple addressing is intended specifically for those cases where there are multiple devices or addressable device functions within one mainframe. A block of secondary addresses is reserved by the device and these are assigned to the various T and L functions as the designer may choose.

While it is possible to mix devices which use simple addressing, extended addressing, and multiple addressing on the same loop, this should only be done with a good deal of caution and thought for the interactions of the various addresses. For example, the controller will need to assign addresses to the devices with extended addressing before assigning addresses to multiple addressing devices. Also, devices may be built that can perform either two byte addressing or one byte addressing (depending on the order that the controller assigns them). The controller must also exercise care such that no two talker functions are assigned the same primary and secondary address combinations (no secondary address is equivalent to having all secondary address codes assigned).

5.4 Asynchronous Operations

The normal loop handshake allows for only one frame to be in transit around the loop at any given time. For three special tasks, however, the controller may choose not to use this standard message handshake. These cases are: loop power-up or reset; integrity check or SRQ detect during slow handshake cycles; SRQ detect during quiescent periods.

Due to the serial nature of the loop and the presence of devices of various speeds, multiple frames may "stack up" at one device and frames may be lost if the normal loop handshake is not performed. For this reason, it is important to view asynchronous loop operation as a separate, non-standard mode of operation in which only certain, carefully defined tasks are accomplished. The burden of handling the possibilities mentioned here rests with the active controller and only the active or system controller may enable or disable the asynchronous modes of operation. This may affect the controller designer, the system programmer, or both.

5.4.1 Power up and Error Recovery

If the system controller chooses to power up the
interface or reset it from an error condition, it may do so by
sourcing the IFC (interface clear) message. Only the system
controller may send the IFC message. It may do so at any time it
chooses with the understanding that it may cause data loss if
performed during data transfers.

Generally, sourcing the IFC is an asynchronous operation
and should always be performed in the following manner. First,
the system controller should send the IFC message and then wait
for it to return. If it does not return within some relatively
long time period (100 milliseconds, for example), the system
controller should send the IFC again and wait, repeating this
sequence until the IFC returns. It is possible that other frames
will be received by the system controller before the IFC returns.
Because these frames may have been sourced by a device that has
become idle (due to the IFC message), they must be destroyed by
the system controller to prevent their endless circulation around
the loop. When an IFC returns, the system controller should then
source one and only one RFC message to complete the handshake.
All frames received by the system controller before the RFC
returns must also be destroyed. When the RFC returns the system
controller knows that all interface functions are initialized and
that the loop is free of extraneous frames. If the RFC message
does not return after a long time (say, 10 seconds), an error has
occurred and the entire sequence should be repeated.

The IFC sequence is not intended to preserve message
frames on the loop so the system controller need not concern
itself with the destroyed frames. All system controllers must be
able to execute this sequence in order to properly power up the
interface.

5.4.2 Loop Integrity Check

During very slow loop operations, the controller may
source an IDY (identify) message asynchronously to verify loop
integrity or to check for service requests (for improved response
time). The IDY is immmediately retransmitted by all devices and
will return to the controller relatively rapidly with service
request and, optionally, parallel poll information. If the IDY
message does not return within a reasonable amount of time, then
the loop may be broken and an error condition exists.

The active controller may send the asynchronous IDY
during any loop operation, but great care must be exercised
during the RFC handshake of the CMD-RFC sequence. Since the CMD-
RFC handshake cycle is the only case in which loop devices will
normally have two frames waiting for interpretation (the CMD and
the RFC), asynchronous IDY frames sent before the RFC frame
returns risk destroying the RFC or the IDY frame (or both)
because of frames stacking up. Therefore, IDY frames should not
be sent by the active controller after RFC frames unless error
recovery techniques (such as sending the RFC again) are used.

In cases other than the CMD-RFC sequence discussed above,
the asynchronous IDY frame will not cause frames to be lost and
the IDY will very likely overtake and pass whatever other message
is presently on the loop. The functional specifications require
that this other message not be affected in any way by the passage
of the IDY. Loop operation should continue without any loss of
data. Provided that the controller only sources one asynchronous
IDY at a time, there will be no problems with frames being lost
by the loop devices. When the IDY returns, the controller is
then free to source another IDY if desired.

5.4.3 Asynchronous Service Requests

When the loop is quiescent, devices do not have a way to
notify the controller of requests for service. Therefore, for
the controller to know when service has been requested, it must
either periodically send IDY frames, or it may enable devices to
source their own asynchronous IDY frames (with the SRQ bit set).

To enable devices to source their own IDY frames, the
controller sends the EAR command (enable asynchronous requests),
the RFC, and then simply stops sending messages. When devices
receive this command, they enter a mode in which, if they have
the capability and require service, they may source their own
asynchronous IDY. This message will not be retransmitted by the
active controller, but may cause it to resume normal operation
which includes handling the service request.

Because of the possibility of destroyed frames, the
controller may only source UCG (universal command group) messages
while the asynchronous request mode is in effect, and all UCG
messages (except EAR and LPD) disable this mode and return the
loop to normal operation. Note that parallel poll information
contained in IDY messages from other devices is not accurate
since they may not have travelled completely around the loop.

Devices are enabled to source their own IDY frames immediately after receiving the EAR command, and therefore, the controller must not require the RFC following the EAR command to return (it could be destroyed by the IDY's).

The controller should return the loop to normal operating mode in the same manner as the system controller handles the IFC sequence due to the asynchronous nature of this mode. The UCG command is sent first which disables asynchronous request mode. NOP is recommended as it does not otherwise modify the state of the interface (other UCG commands may be used if their effects are desirable). Other IDY messages may return before the CMD is received. They should be ignored (not retransmitted). If the UCG command does not return after a period of time, it should be sent again, and so on, until it does return (there is a possibility that the UCG command could be destroyed by frame "stack up"). The RFC can then be sent. One or more additional IDY messages may possibly arrive before the RFC returns. They should be ignored as were the earlier ones. When the RFC returns, the loop is cleared and has been returned to its normal, synchronous operating state. The controller can now perform normal operations such as servicing the request. The same sequence should be used to bring up a loop which has been powered down with the LPD command.

It is easily seen that the implementation and programming of controllers with asynchronous capabilities is not a simple task. It should only be attempted by designers and programmers who have a complete and total understanding of loop protocol.

5.5 Operational Sequences

To understand more completely the operation of an HP-IL system, frame sequences of several typical operations are shown. The sequences are not intended to be exhaustive, nor do they necessarily represent the only way to accomplish the given task. They are included as recommended sequences for performing the given operation. The messages listed in the column on the left represent those sent by the controller. The column on the right shows the messages sourced by the talker.

Initial System Power On

IFC	The system controller initially sends the IFC
IFC	(Interface Clear) command at regular, slow intervals
IFC	stopping any and all frames received until an IFC
.	returns. When an IFC frame returns, the controller
.	knows that all devices are powered up and are properly
IFC	sending and receiving frames. The system controller
	then sends one and only one ready for command message
RFC	to complete the handshake and waits for it to return
	also stopping any other frames received. When the RFC
	returns, the controller knows that the loop devices
	have properly completed interface initialization and
	that no extra frames are on the loop. If the RFC does
	not return, the sequence should be repeated.

Talk-only, Listen-only System Power On

DAB1	The talker immediately sends its first data byte and
.	continues to send it at slow, regular intervals until
.	it returns, indicating that the other devices on the
DAB1	loop are powered up and are properly retransmitting or
	receiving frames. The talker then sends the rest of
DAB2	its data at normal speed. No EOT message is sent in
DAB3	a talk-only system since it would circulate endlessly.
.	Transmission continues in this manner until the local
DABn	message ton is turned off.

Data Transfer

UNL		The controller first inhibits any previous listeners
IFC		with the UNL (Unlisten) command and enables a listener
LADn		with the appropriate listen address command. More
RFC		than one listener can be enabled, if desired. The
TADn		talk address command disables any previous talkers and
RFC		enables one device to send data when it receives the
	SDA	proper ready message. The controller now sends the
	DAB1	Send Data ready message which the talker replaces on
	DAB2	the loop with its first data byte. As each byte
	DAB2	returns, the talker sends the next data byte of its
	.	message. When the talker has no more data ready to
	.	send, it sources the ETO (End of Transmission, OK)
	.	message. ETO indicates to the controller that all
	.	data frames sourced by the talker returned without
.		error. The controller replaces the ETO message with
.		its next interface message.

Interruption of Data Transmission

	Assume the talker is sending its seventh data byte.
	The device halting the data transfer holds the
DAB7	talker's byte, and replaces it with the NRD (Not Ready
NRD	for Data) message. The NRD signals the talker to stop
NRD	its data transmission immediately. The talker returns
DAB7	the NRD frame to the halting device which retransmits
ETO	the held data byte. When the talker receives its data
.	byte, it sends an end of transmission message to allow
.	the controller to perform other operations (such as
.	handling a service request). If the talker is again

directed to send data, the transmission will continue
at the point of interruption (unless directed
otherwise in a device dependent manner).

Serial Poll

UNL	The controller first sends the unlisten command to
RFC	disable all previous listeners.
TAD1	
RFC	The first device is addressed to talk and asked to send
SST	its status with the Send Status ready message. The
DAB	talker replaces the SST frame with its status message
.	(one or more bytes of data) followed by ETO.
ETO	
TAD2	The controller continues by addressing the second
RFC	device to talk. This same sequence (TAD,RFC,SST) is
SST	repeated for each device on the loop. Devices that
DAB	currently have a need for service will respond to the
.	serial poll with bit 6 of the first status byte set.
ETO	If the SST frame returns to the controller, it knows
TAD3	that the addressed device does not have the capability
.	to request service or to respond with its status.

Control Passing

TAD	The currently active controller sends the talk address
RFC	of the device to which it will pass loop control
	followed by the TCT (Take Control) ready message.
	When the TCT message is received by the talker, itive
TCT	takes over active control of the loop by replacing the
.	TCT with its first interface message. If, instead,
.	the TCT returns to the previous controller, the talker
.	has not assumed control of the loop and the previous
	controller must resume control.

Parallel Poll Configuration and Operation

PPU   The controller first may send the PPU (Parallel Poll
RFC   Unconfigure) command to disable all previous parallel
         poll response assignments.

UNL   The controller sends the Unlisten command to prevent
RFC   unwanted devices from reacting to parallel poll enable
LADn   commands. The listen address of the device to be
RFC   configured is sent next followed by the specific
PPE13   parallel poll enable command. In this case, the device
RFC   is assigned to respond with a positive response by
         ORing a 1 into bit 3 of an IDY frame.
UNL
RFC   This same sequence (UNL,RFC,LAD,RFC,PPE,RFC) is
 .   repeated for each device to be configured for parallel
)   poll. Individual devices can be disabled by
 .   substituting the PPD (parallel poll disable) command
         for the PPE command in this same sequence.

IDY0   After the devices have been configured, the controller
         may execute the parallel poll at any time by sending
         the IDY message. The frame will return with its data
         bits set according to the configuration commands and
         the devices' individual status bits.

Assign One Byte Addresses

AAU   The controller first sends the AAU (Auto Address
RFC   Unconfigure) command to reset any previous address
             assignments and make all devices ready to receive new
 AD1   addresses. Then the controller sends the AAD (Auto
             Address) ready message. The first loop device accepts
             the lower five bits of the AAD message as its new
             address assignment and increments the AAD message
             before sending it on to the next device. This process
             (accept address, increment, send to next device)
             occurs at each device on the loop until the modified
     AADn   frame returns to the controller, indicating the number
             of devices on the loop. If the frame returned with
             address 31, there may be too many devices on the loop.
             To determine this, the controller should now send
             AAD30. If it returns unchanged, there are exactly the
             maximum number of devices. If IAA returns, there are
             too many devices and improper operation may result.

)

Appendix A. Capability Subsets

R (Receiver) Interface Function

All devices must have the complete R function implemented.

D (Driver) Interface Function

All devices must have the complete D function implemented.

AH (Acceptor Handshake) Interface Function

All devices must have the complete AH function implemented.

SH (Source Handshake) Interface Function

All devices must have the complete SH function implemented.

C (Controller) Interface Function

Identification	Requirements
C0 No capability	Omit all states
C1 Basic capability	Implement CIDS, CACS, CSBS, CEIS, CEMS
2 System controller	Additional to C1; local messages sic, scl not always false
3 Respond to service requests	Additional to C1; add CSNS, CSRS (needs listener capability)
4 Pass, receive control	Additional to C1; add CTRS and optional term; (requires talker capability)
5 Parallel poll	Additional to C1
6 Enable and interpret asynchronous IDY frames	Additional to C1

A-1

T (Talker) Interface Function

Identification	Requirements
TØ No capability	Omit all states
T1 Send data	Implement TIDS,TADS,TACS,TAHS,TERS
T2 Send status	Implement TIDS,TADS,SPAS,TAHS,TERS
T3 Send device ID	Implement TIDS,TADS,DIAS,TAHS,TERS
T4 Send accessory ID	Implement TIDS,TADS,AIAS,TAHS,TERS
5 Talk-only mode	Additional to T1; local message ton not always false
6 Extended talker	Additional to T1-4; implement TPIS and TPAS (requires AA2 or AA3)

Talker capability consists of one or more of T1-4.

L (Listener) Interface Function

Identification	Requirements
LØ No capability	Omit all states
1 Basic capability	Implement LIDS, LACS
2 Listen-only mode	Additional to L1; local message lon not always false
3 Extended listener	Additional to L1; implement LPIS and LPAS (requires AA2 or AA3)
4 Not ready	Additional to L1; implement NIDS, NENS, NRWS, NACS

SR (Service Request) Interface Function

Identification	Requirements
SRØ No capability	Omit all states
SR1 Basic capability	Implement SRIS, SRSS, SRHS; needs T2 capability
SR2 Basic and asynchronous request capability	Implement all states; requires T2 capability

Appendix A Capability Subsets

RL (Remote Local) Interface Function

Identification   Requirements

RLØ No capability   Omit all states
RL1 Basic capability   Implement RIDS, RACS, LOCS,
                                    and REMS; local message rtl
                                    always false; needs
                                    listener capability
RL2 Basic capability with   Implement all states; local
       local lockout   message rtl not always
                                    false; needs listener
                                    capability

AA (Auto Address) Interface Function

Identification   Requirements

AA1 Basic capability   Implement AAUS, AAIS, AACS
AA2 Extended addressing   Implement AAUS, ASIS, AWPS, AECS
AA3 Multiple addressing   Implement AAUS, APIS, AWSS, AMIS,
                                    AECS
All devices must implement one or more of AA1-3.

PD (Power Down) Interface Function

Identification   Requirements

PDØ Basic capability   Implement POFS, PONS, PUPS;
                                    pseudomessage edge always false
PD1 Complete capability;   Implement all states
       responds to power
       down command

All devices must implement one of PDØ, PD1.

)

PP (Parallel Poll) Interface Function

Identification   Requirements

PP0   No capability   Omit all states
PP1   Complete capability   Implement all states; needs
                                      listener capability

DC (Device Clear) Interface Function

Identification   Requirements

)C0   No capability   Omit all states
DC1   Respond to universal   Implement all states; omit
         device clear command   optional term
DC2   Respond to universal   Implement all states; include
         and addressed device   optional term; needs listener
         clear commands   capability

DT (Device Trigger) Interface Function

Identification   Requirements

DT0   No capability   Omit all states
DT1   Complete capability   Implement all states; needs
                                      listener capability
)

DD (Device Dependent Commands) Interface Functions

Identification   Requirements

DD0   No capability   Omit all states
DD1   Responds to one or   Implement all states; needs
         more device dependent   listener capability for DDL,
         listener or talker   talker capability for DDT
         commands   commands

A-4

)

Appendix B. Message Glossary

B.1 Local Messages and Pseudomessages

arq — Asynchronous request. To SR (service request) function.
The device uses arq to signal the function to send the
asynchronous IDY (identify) message to the controller to
request service. This will only happen if the device has
this capability and has been enabled to do this by the EAR
(enable asynchronous requests) command. The arq message
must go false as soon as the function enters the state
which sources the IDY (ARSS). It may go true again upon
return to the standby state (SRSS).

edge — Wake-up signal. Pseudomessage to PD (power down)
function. If the device has the full power down
capability and has been powered down with the LPD (loop
power down) command, the function will be in the power off
state, even though the power switch will still be on. In
this condition, any frame (actually, any pulse) on the
loop will generate the edge message and cause the function
to bring the device back to full power. The edge message
is a pulse which must only last long enough for the device
to completely power on to its initial state. When edge
goes false, the device will then be ready for normal
operation.

fon — Force on. To PD (power down) function. Under certain
conditions, devices which have the full power down
capability may need to remain powered up even though they
have received the LPD (loop power down) command. The fon
local message causes them to go back to PUPS (power up
state) rather than go to POFS (power off state). The
active controller would use this feature to remain awake,
while presently inactive controllers could allow
themselves to power down normally. If a device was
enabled to send asynchronous IDY messages to wake up the
loop, it might also use this message to remain awake.

)fre - Frame error detected. To T (talker) function. All data
        frames sourced by the active talker should be error
        checked when they return. If an error is detected, then
        the fre message should be sent to the T function causing
        it to source the ETE (end of transmission, error) message.

frtc - Frame transmission complete. Pseudomessage to D (driver)
        function. This message is generated by the encoder
        circuitry to signal the D function that it is done
        transmitting an entire frame and that the function can now
        return to the idle state to wait to begin transmitting
        another frame. This message should only be true until the
        function returns to the idle state.

gta - Go to active. To C (controller) function. When the
        controller enables a device dependent transmission, it
        goes to the standby state to wait for completion of the
        transmission. While the controller is in its standby
        state, it may not source any frames. If it becomes
        necessary to source asynchronous frames such as IDY or
        IFC, the device may force the C function to become active
        (so that it is enabled to source frames). The gta message
        should only be true until the function returns to the
        active state.

gts - Go to standby. To C (controller) function. After the
        controller has used the gta message to force the C
        function to become active during a device dependent
        transmission, it uses the gts message to return to the
        standby state to wait for the normal completion of the
        transmission. This message should only be true until the
        function returns to the standby state.

)hlt - Halt data transfer. To L (listener) function. This
        message is used by the device to cause the L function (if
        controller or enabled by the controller) to send an NRD
        (not ready for data) sequence to halt the data transfer.
        The hlt message is especially useful for listeners with
        limited input buffers.

lab - Local abort. To SH (source handshake) function. This
        message is used for asynchronous operations to cause the
        SH function to abort the current handshake and become
        ready for another message. It would otherwise be "hung"
        waiting for the previous message to return before it could
        generate the new message. The lab message is only true
        until the function returns to the generate state.

lfs - Last frame sent. To the T (talker) function. The device
       will send this message true when it detects that there is
       no more data ready to be sent. The T function will then
       source the ETO (end of transmission, OK) message unless
       the fre local message is also true.

lon - Listen only. To L (listener) function. This message is
       usually controlled by a manual switch and serves to make
       the device an active listener even though it has not been
       addressed. This is only allowed in a special system
       configuration which has no controller called a talk-only,
       listen-only system.

ltn - Local listen. To L (listener) function. This message is
       used by the active controller to make itself an active
       listener without sending out its own listen address,
       though it could do this if it wished. This message might
       be used to allow the controller to monitor a device
       dependent message transfer between other devices or it
       might also be used if the controller itself were the
       destination of the data. The ltn message will only be
       true until the function enters the active state.

lun - Local unlisten. To L (listener) function. This message
       is used by the active controller to return the listener
       function to the idle state after it has been forced active
       by the ltn local message. This message will only be true
       until the function returns to the idle state.

nfa - New frame available. To SH (source handshake) function.
       When nfa goes true, the device is telling the SH function
       that it has generated another frame for transmission and
       is ready to have it sent. The nfa message must go false
       before the function can become ready to source the next
       frame.

pof - Power off. To PD (power down) function. This message is
       usually sent true by the "off" position of the device
       power switch. It causes an immediate transition from any
       state to POFS (power off state). This, in turn, causes
       all other functions to go to their power off states. This
       message remains true until the switch is moved to the "on"
       position.

)

pon - Power on. To PD (power down) function. This message is a
       short pulse generated when the device power switch is
       moved to the "on" position. It causes the PD function to
       move from the power off state to PONS (power on state).
       The function remains in PONS until pon goes false. The
       pon message must last long enough to allow all device and
       interface functions to go from powered down to powered up
       and ready for normal operation. When pon goes false, the
       function enters PUPS (power up state).

rdy - Ready. To AH (acceptor handshake) function. With this
       message the device indicates to the AH function that it is
       now ready to receive the next byte of the incoming message
       string. The rdy message allows transition to ACDS
       (acceptor data state) which transfers incoming frames to
       all the other interface functions and device functions.
)   The device indicates acceptance of the message from the
       function by setting rdy false. As long as the device is
       busy (it is interpreting the current message or its buffer
       is full) the rdy message should remain false. This
       message, therefore, controls the handshake of messages
       into the device.

rsv - Request service. To SR (service request) function. The
       device indicates to the SR function a need for service
       with this message. It causes the function to begin
       sending the SRQ (service request) to the controller when
       it has the opportunity. The rsv message is the same as
       bit D6 of the device's first status byte.

rtl - Return to local. To RL (remote local) function. This
       message is usually generated by a button on the device
)   which returns control of the device functions to manual
       controls on the instrument itself. The function has the
       optional capability to ignore this message with the LLO
       (local lockout) command. The message must not be always
       true; it will usually be a short pulse.

scl - System controller. To C (controller) function. This
       message indicates to the function that it is the system
       controller, the only device permitted to send the IFC
       (interface clear) message to take control of the loop at
       any time. This message should remain true in one and only
       one device throughout the operation of the interface.

)

sic - Send interface clear. To C (controller) function. The
       system controller uses this message to cause the function
       to go from the idle state to the active state so that it
       can send the IFC (interface clear) message and take
       control of the loop. The sic message must be used at
       initial power on as well as any other time the system
       controller needs to asynchronously take control of the
       interface. The message should only be true until the
       active state is entered.

sync - Valid sync bit received. Pseudomessage to R (receiver)
       function. This message is generated by the decoder
       circuitry when it recognizes the beginning of a message
       frame from the loop. It causes the function to go from
       idle to RSYS (receiver sync state) where it will decide
       whether the message is for this device or not, and then
       take appropriate action. The sync message should only be
       true until the function enters RSYS.

tlk - Local talk. To T (talker) function. With this message,
       the active controller can make itself the active talker
       and source device dependent data without sending an SDA
       (send data) message although it could do this instead, if
       desired. The tlk message should only be true until TACS
       (talker active state) becomes true.

ton - Talk only. To T (talker) function. This message is
       normally controlled by a manual switch and causes the
       device to become the active talker even though it has not
       received its talk address. This is only permitted in a
       special system configuration which lacks a controller and
       is called a talk-only, listen-only system. The function
       will remain in TACS (talker active state) sending
       continuous data messages as long as the switch is in this
       position.

B.2   Remote Messages

AADn - Auto address n. Auto address group, ready class,
       101 100aaaaa. The controller sends AADn to assign simple
       (one byte) addresses to devices on the loop. The lower
       five bits represent a binary coded address number n which
       can range from 0 to 30 (31 is an illegal address, devices
       will not respond to this value). Each device accepts the
       incoming value n as its address, increments this value,
       and sends the modified AAD to the next device on the loop,
       which, in turn, does the same. Once a device has received
       its address in this way, it will no longer respond to any
       AAG (auto address group) message until after the AAU
       command is received or the device is powered off, then on
       again. The controller uses the address value which
       returns after going through each device around the loop to
       determine the number of devices, or to determine if there
       are too many devices.

AAG - Auto address group. Ready class, 101 1xxxxxx. This
       mnemonic indicates the entire group of auto address ready
       frames, including simple address, extended and multiple
       address, and secondary address assignment frames. The
       controller uses these to assign addresses to devices on
       the loop in various ways.

AAU - Auto address unconfigure. Universal command group,
       command class, 100 10011010. The controller uses this
       command to cause all devices to reset their address
       assignments. After an AAU, devices must respond to either
       address switches or a preset address. If loop devices
       already have addresses assigned, the controller must send
       the AAU message before assigning new addresses to those
       devices. Note that IFC (interface clear) does not affect
       address assignments in any way.

ACG - Addressed command group. Command class, 100 x000xxxx or
       100 101xxxxx or 100 110xxxxx. This mnemonic indicates
       that group of commands to which a device does not respond
       unless it is addressed as a talker or a listener,
       depending on the particular command. This group also
       includes the device dependent commands, DDLn and DDTn, as
       well as others.

AEPn - Auto extended primary n.   Auto address group, ready class,
       101 101aaaaa. After the controller has assigned secondary
       addresses with the AESn message to a group of devices
       which can accept extended addresses, it uses the AEPn
       message to assign the same primary address to each device
       in the group. Devices do not modify this message, they
       merely accept the address assignment and send the message
       to the next device. Other devices do not respond to this
       message. After the AEP, the device is configured and can
       respond to its assigned secondary and primary addresses.
       AAU will reset the address assignment and ready the device
       to receive a new address. aaaaa represents the five bit
       binary address n, which can range from 0 to 30 (31 is an
       illegal address, devices will not respond to 31).

AESn - Auto extended secondary n. Auto address group, ready
       class, 101 110aaaaa. AES is used by the controller to
       assign secondary addresses to extended addressable
       devices. The lower five bits contain the binary address
       n, which can range from 0 to 30 (31 is illegal). Each
       device accepts the value n as its secondary address,
       increments this value, and sends the modified message on
       to the next device. When the value reaches 31, no other
       devices respond and the message simply returns to the
       controller. The controller can then use AEPn to assign
       the primary address to this group of devices. Once
       configured, the devices can no longer respond to the AESn,
       so the controller can now send it out again to assign
       extended addresses to the next group of devices on the
       loop. The primary address for each group must, of course,
       be unique.

AMPn - Auto multiple primary. Auto address group, ready class,
       101 111aaaaa. AMP assigns primary addresses to all
       devices which use multiple addressing on the loop. The
       lower five bits represent a binary address n, which can
       range from 0 to 30 (31 is illegal). The controller sends
       the AMP message and each succeeding device accepts the
       incoming value as its new address, increments n, and sends
       the message to the next device, which, in turn, does the
       same. The value which returns to the controller indicates
       the number of multiple address devices on the loop.
       Following this, the controller sends the ZES command to
       each device so that it can reserve the proper sized block
       of secondary addresses. The device is then configured and
       can respond to its assigned addresses. AAU is necessary
       before devices will respond to new address assignments.

ARG - Addressed ready group. Ready class, 101 01xxxxxx. Only
       talkers, listeners, and controllers may respond to this
       group of messages. Idle devices must ignore (retransmit)
       these messages. ARG messages include SOT (start of
       transmission), EOT (end of transmission), and NRD (not
       ready for data) subgroups and messages. With the
       exception of NRD, these messages do not normally travel
       all the way around the loop back to the sourcing device.
       In general, they serve a handshake function and the
       destination device replaces them with another message. At
       present, listeners do not respond to these messages, but
       may source the NRD message if enabled by the controller.

CMD - Command. 100 xxxxxxxx. Commands are one of the major
       classes of loop messages. They control the operation of
       the interface functions of each device in a major way, and
       to a lesser extent, the device functions also. The active
       controller is the only device which may source command
       messages (except for asynchronous IFC by the system
       controller). Every command must be immediately followed
       by the RFC message to provide devices the oportunity to
       handshake, that is, to indicate they are ready to receive
       the next command. Commands are immediately retransmitted
       by all devices to minimize delay but a copy of the message
       is saved by each device to begin execution of the command.

DAB - Data byte. Data or end class, 00x xxxxxxxx. Data bytes
       are the basic unit of the device dependent message
       transmission. This is the data which the interface system
       is designed to handle. The other messages are largely
       overhead for control purposes. These messages between
       devices may be coded in any way but it is strongly
       recommended that ASCII be used wherever possible for
       compatibility reasons. The data byte also contains the
       SRQ bit (C0) which devices may set to indicate to the
       controller a need for service.

DCL - Device clear. Universal command group, command class,
       100 00010100. DCL is sent by the controller to cause all
       devices which recognize this command to set their device
       functions to a preset state, whether they are addressed or
       not. This command does not affect the interface
       functions. The preset state is defined by the device
       designer and is normally the same as the power-on state.

DDLn - Device dependent listener command n. Addressed command
        group, command class, 100 101xxxxx. A device must be
        addressed as a listener in order to respond to any one of
        the 32 possible DDL commands. The particular effect of
        the specific command is designer determined but it must
        not directly affect any interface functions.

DDTn --Device dependent talker command n. Addressed command
        group, command class, 100 110xxxxx. A device must be
        addressed as a talker in order to respond to any one of
        the 32 possible DDT commands. The particular effect of
        the specific command is designer determined but it must
        not directly affect any interface functions.

DOE   - Data or end. 0xx xxxxxxxx. This major frame
        classification includes all the device dependent messages
        for the interface system. This is the data which is
        communicated from one device to another and for which the
        system was designed. DOE frames include the END bit (C1)
        to indicate an end-of-record condition without terminating
        the transmission, and the SRQ bit (C0) for devices to
        indicate a need for service to the active controller.
        These messages are sourced by the active talker and are
        received by the active listener(s) on the loop.

EAR   - Enable asynchronous requests. Universal command group,
        command class, 100 00011000. This command is used by the
        active controller to put all devices which have the
        capability in a mode where they can source their own IDY
        message (with service request bit set) to indicate a need
        for service to the controller. Normally, the controller
        is the only device to source IDY frames. After the
        controller sends the EAR - RFC sequence, it will allow the
        loop to go idle, until such time as an asynchronous IDY
        from one of the devices arrives or until the controller
        must perform some other operation. To disable the
        asynchronous request mode, the controller must send out a
        universal command which disables the asynchronous request
        mode and then resume normal operation. All universal
        commands except EAR and LPD disable the mode. However, it
        is recommended that controllers use the NOP command as it
        has no affect on the other interface functions. LPD does
        not disable the mode so that it is possible to have some
        device other than the active controller wake up a powered
        down loop.

) ELN - Enable listener not ready. Addressed command group,
        command class, 100 00001111. This command enables
        listeners to halt data transfers when necessary. Any
        device that is active to listen may respond to the EDN
        command.

  END - End data byte. Data or end class, 01x xxxxxxx. The end
        byte is the same as a data byte except that bit C1 is set
        to indicate an end-of-record condition to the listener.
        This has no affect on the interface functions and the end
        byte is treated exactly the same as any other data byte.
        Most ASCII transmissions will indicate end-of-line with a
        CR, LF pair, for example, but binary data will probably
        need to use the END message for this function. The END
        byte does not terminate the transmission.

) EOT - End of transmmission. Addressed ready group, ready class,
)   101 0100000x. This is a subgroup including two messages,
        ETE (end of transmission with error) and ETO (end of
        transmission, OK). The active talker sources these
        messages to indicate the end of a data transfer to the
        controller. The controller replaces the EOT message with
        the next interface message. Listeners must ignore
        (retransmit) the EOT messages.

  ETE - End of transmission with error. Addressed ready group,
        ready class, 101 01000001. This message is sourced by the
        active talker to indicate to the controller that a data
        message sent by the talker has returned in error. This
        error checking capability is strongly recommended but not
        required. If the device does not perform error checking,
        it may not source this message. The controller replaces
        this message with its next operation on the loop, possibly
)   an attempt to restart the transmission.

  ETO - End of transmission, OK. Addressed ready group, ready
        class, 101 01000000. This message is sourced by the
        active talker to indicate to the active controller that it
        is no longer actively sourcing data. Though not required,
        error checking is strongly recommended. If error checking
        is not performed, the talker must assume that no errors
        have occurred and end its data transfer with the ETO
        message. The ETO message does not return to the talker
        but instead the controller replaces it with the next
        interface message. Active listeners must ignore
        (retransmit) the ETO message.

)

GET   - Group execute trigger. Addressed command group, command
        class, 100 00001000. GET is a command used by the
        controller to cause all devices which are listener
        addressed to begin their particular device operation.
        This operation for each device is designer specified. The
        controller may use the GET command to start an operation
        in several devices as nearly at the same time as is
        possible given the loop architecture of the system.

GTL   - Go to local. Addressed command group, command class,
        100 00000001. The controller uses this command to put all
        devices which are listener addressed under control of
        their local (front panel) controls. Programming data for
        the device will, in general, be ignored while the device
        is in this state, if it is received from the interface.

IAA   - Illegal auto address. Auto address group, ready class,
        101 10011111. If the controller receives this message as
        a result of assigning addresses with the AAD message,
        there may be exactly the maximum or too many devices on
        the loop. Since devices do not respond to the IAA
        message, one or more devices may not have been assigned an
        address. To test whether all devices have been assigned
        an address, the controller should send the AAD30 message.
        If AAD30 is returned, then exactly the maximum number of
        devices are on the loop and the controller may begin
        normal operations. If IAA is returned again, then too
        many devices are present and more than one device may
        respond to a particular address. Before normal operations
        may begin, all extra devices must either be removed from
        the loop or assigned to addresses from an unused address
        range. For example, the extra devices can all be assigned
        address 30 by repeatedly sending AAD30 until returns
        unchanged.

IDY   - Identify. 11x xxxxxxx. This major classification of
        messages is used by the controller to perform parallel
        poll or to check for service request. Bit C0 is set by
        devices which need service. If the controller has
        configured devices to respond to parallel poll, these
        devices set designated data bits in the IDY message as it
        passes through the device. With this capability, the
        controller can rapidly identify which device needs
        service. During normal operations IDY messages may only
        be sourced by the active controller, but other devices can
        be enabled to send IDY frames (with the service request
        bit set) with the EAR command.

IEP - Illegal extended primary. Auto address group, ready
       class, 101 10111111. This message is this same as the AEP
       message except that the address value is 31, an illegal
       value. Devices will not respond to this message.
       Furthermore, since AEP is not incremented by the loop
       devices, it will not be received by the controller and it
       is included here only for consistency.

IES - Illegal extended secondary. Auto address group, ready
       class, 101 11011111. Extended address devices receive
       their secondary address assignments via the AES message,
       which they accept, increment, and send on to the next
       device. When the address value reaches 31, it is defined
       as the IES message and is no longer accepted by other
       devices, which pass it unchanged back to the controller.
       The controller then assigns primary addresses with the AEP
       message, which is not incremented. Only devices which
       have received the AES message (and have not yet received
       the AEP message) will respond. After receiving both
       secondary and primary addresses, devices will no longer
       respond to AAG messages and the controller may configure
       the next group of devices. If the controller generates
       the IEP message internally and the AES message returns
       incremented, there are too many devices on the loop.

IFC - Interface clear. Universal command group, command class,
       100 10010000. IFC may be sourced only by the system
       controller. It may be sent at any time to take control of
       the interface system. IFC resets all talker, listener,
       and controller functions on the loop to their idle state,
       but does not affect any other interface or device
       functions. IFC also must not affect the parallel poll or
       address assignment.

IMP - Illegal multiple primary. Auto address group, ready
       class, 101 11111111. The controller uses the AMP message
       to assign primary addresses to those devices which have
       multiple address capability. If there are exactly the
       maximum number or too many devices of this type on the
       loop, the AMP message will be incremented to 31, which is
       defined as the IMP message. No further devices will
       respond and the message will return to the controller.
       The controller should send AMP30 at this point. If it
       returns unchanged, the loop has exactly the maximum number
       of devices. If it returns modified, there are too many.
       The controller should signal an error condition and not
       attempt normal operations.

LADn - Listen address n command. Listen address group, command
       class, 100 001aaaaa. This is the command that the
       controller uses to cause a particular device to become the
       active listener, that is, able to receive and interpret
       data messages from the loop. The lower five bits
       represent a binary address n, which can range from 0 to
       30. 31 is an illegal address which causes all listeners
       to go to the idle state. It is called the unlisten
       command. For devices which use a two byte address, LAD
       provides the primary address only. The MSA command must
       be received to make these devices active listeners.
       Multiple LAD messages will activate multiple listeners.

LAG - Listen address group. Command class, 100 001xxxxx. This
       group of commands includes all the listen address commands
       and also the unlisten command (address 31). Secondary
       address commands are in a separate group; only primary
       listen addresses are included in LAG.

LLO - Local lockout. Universal command group, command class,
       100 00010001. With this command the controller can cause
       all devices which respond to this command to lock out, or
       not respond to, their return to local control buttons on
       the instruments. This will prevent an operator from
       changing a device's control settings inadvertantly at a
       critical time.

LPD - Loop power down. Universal command group, command class,
       100 10011011. The controller uses this command to place
       the loop, or rather all devices which respond to this
       command, in a power-down state to conserve power. The
       controller remains powered up so that it can wake up the
       loop at a later time and continue normal operations. If
       the controller has enabled the asynchronous request mode
       prior to sending the LPD, other devices with the proper
       capability can also initiate the loop wake-up sequence.

MLA - My listen address. Listen address group, command class,
       100 001mmmmm. This is the particular LADn command which
       happens to match in the least significant five bits with
       the address (primary address in the case of devices that
       have a two byte address) which is assigned to this
       specific device. This command causes the device to become
       the active listener and able to receive device dependent
       messages. In the case of devices which require two byte
       addresses, the MSA code is also required before the device
       becomes active.

MSA - My secondary address. Secondary address group, command
       class, 100 011mmmmm. This is the particular SADn command
       which matches the secondary address assigned to this
       specific device. This command causes the device to become
       the active talker or to become an active listener. This
       command must immediately follow the MTA or MLA command.

MTA - My talk address. Talk address group, command class,
       100 010mmmmm. This is the particular TADn command which
       matches the talk address (primary address in the case of
       devices that have a two byte address) which is assigned to
       this specific device. When the device receives MTA, it
       becomes the active talker on the loop, and, when enabled,
       will source device dependent data on the loop. For
       devices which have a two byte address, the MSA command is
       also required before the device becomes addressed to talk.
       This command causes the previous talker to become idle.

NAA - Next auto address. Auto address group, ready class,
       101 100nnnnn. This mnemonic represents the incremented auto
       address message that a device sends on to the next device
       on the loop. The value of the lower five bits might range
       from 1 to 31, depending on the value of the AAD message
       before incrementing. The value 31 is also called IAA,
       values less than 31 are also called AAD.

NES - Next extended secondary. Auto address group, ready class,
       101 110nnnnn. The NES frame is the incremented secondary
       address assignment which devices send to the next loop
       device. It is used by devices with extended or multiple
       address capability. The address bits (nnnnn) can range
       from 1 to 31, depending on the received AES message.
       Values of 1 to 30 are also called AES and value 31 is IES.

NMP - Next multiple primary. Auto address group, ready class,
       101 111nnnnn. This mnemonic represents the incremented
       primary address message that multiple address devices send
       on to the next device on the loop. The lower five bits
       can range from 1 to 31 depending on the value of the
       received AMP message. The address value 31 is also known
       aso IMP and value less than 31 are also called AMP.

NOP - No Operation. Universal command group, command class,
       100 00010000. This is the universal no operation command.
       It is useful for disabling asynchronous request mode since
       any universal command disables it and the NOP command has
       no other effects.

NRD - Not ready for data. Addressed ready group, ready class,
       101 01000010. When the controller or a device enabled by
       the controller needs to interrupt an active talker during
       a data transmission, it does so by holding the next data
       byte and replacing it with the NRD message. This message
       signals the talker that it should terminate its
       transmission. When the NRD returns to the sourcing
       device, it will then send the held data message. When
       this data message is received by the talker, it sends the
       EOT message to the controller. If an active talker is
       directed to continue the data transfer (with a send data
       message), it must continue at the point of interruption
       unless directed otherwise in a device dependent manner
       defined for that purpose.

NRE - Not remote enable. Universal command group, command
       class, 100 10010011. With this command, the controller
       causes all devices to be placed under local control; that
       is, they will respond to their front panel controls and
       not to programming information received from the loop.
       Devices which do not implement the remote local interface
       function will simply ignore this command.

NUL - Null command. Addressed command group, command class,
       100 00000000. This is the addressed no operation command.
       Devices do not perform any action in response to this
       command so it is useful for such operations as testing the
       loop handshaking.

OSA - Other secondary address. Secondary address group, command
       class, 100 011ttttt. This mnemonic represents any
       secondary address command whose address bits (ttttt) do
       not match the address assigned to this particular device.
       It causes talker functions to become unaddressed.

OTA - Other talk address. Talk address group, command class,
       100 010ttttt. This mnemonic represents any talk address
       command that contains an address (ttttt) that does not
       match the address assigned to this particular device.
       Since there can only be one active talker on the loop at a
       time, this device will return its talker function to the
       idle state.

PPD - Parallel poll disable. Addressed command group, command
       class, 100 00000101. This command is used by the
       controller to cause the devices which are listen addressed
       to no longer respond to parallel polls. The parallel poll
       function returns to its idle state.

)PPEn - Parallel poll enable n. Addressed command group, command
        class, 100 1000sbbb. This command allows the controller
        to configure devices that are addressed to listen to
        respond to parallel polls in various ways. The s bit
        indicates the sense of the device's response. If s is 1
        the device will set the assigned bit of an IDY frame if it
        needs service, if s is 0 the device will set the assigned
        bit if it does not need service. The lower three bits
        (bbb) indicate the binary bit number on which the device
        must respond. 000 indicates bit D0, 001 indicates D1,...,
        111 indicates D7.

  PPU - Parallel poll unconfigure. Universal command group,
        command class, 100 00010101. The controller uses this
        command to disable all devices on the loop from responding
        to parallel polls. The parallel poll function in each
        device will return to its idle state.
)
  RDY - Ready. 101 xxxxxxx. This major class of messages is
        used for several different purposes, including device
        handshake functions and address configuration. Most are
        sourced by the active controller, but some may be sourced
        by other devices under certain conditions.

  REN - Remote enable. Universal command group, command class,
        100 10010010. With this command, the controller enables
        the loop for remote operation. When a device is addressed
        to listen, it enters its remote control state and will no
        longer respond to its front panel controls. Devices which
        do not implement this function will simply retransmit this
        command and take no other action.

  RFC - Read for command. Ready class, 101 00000000. The
)   controller uses this ready frame as the handshake after
        each command so that it knows that all devices on the loop
        have received the command and are ready to receive the
        next. Every command must be immediately followed by the
        RFC message.

  SADn - Secondary address n command. Secondary address group,
        command class, 100 011aaaaa. This command provides the
        secondary address to enable talkers and listeners which
        respond to two byte addresses. The lower five bits
        represent the binary address n, which can range from 0 to
        30. 31 is an illegal address. The secondary address must
        follow the talk or listen address command immediately to
        enable the device to talk or listen.

)

SAG - Secondary address group. Command class, 100 011xxxxx.
       This group of commands contains the secondary addresses,
       that is, SAD commands. These are only used for devices
       which respond to extended or multiple address modes.

SAI - Send accessory identification. Addressed ready group,
       ready class, 101 01100011. This message is used by the
       controller to cause the addressed talker to begin sending
       its accessory ID byte(s). The talker replaces the SAI
       with the accessory ID on the loop, and terminates with the
       proper EOT message, just as in a data transmission. If
       the device does not have accessory ID capability, it
       merely sends the SAI back to the controller to indicate
       indicate that the device cannot respond. Accessory ID
       consists of a single byte whose high order four bits
       indicate the device class (e.g. printer, mass storage,
       etc.) and low order four bits represent the device type.

SDA - Send data. Addressed ready group, ready class,
       101 01100000. This message is used by the controller to
       direct the addressed talker to begin sending its data.
       The talker replaces the SDA with its first byte of data
       and continues to send data until no more is available.
       The talker follows its last data byte with the proper EOT
       message. If the device cannot source data, it merely
       returns the SDA to the controller. If the talker has no
       data ready, it sends ETO.

SDC - Selected device clear. Addressed command group, command
       class, 100 00000100. The SDC command causes active
       listeners to perform their device dependent clear function
       SDC has no affect on other interface functions.

SDI - Send device ID. Addressed ready group, ready class,
       101 01100010. SDI is used by the controller to cause the
       talker addressed device to begin sending its device ID
       string. The talker replaces the SDI with its ID string on
       the loop and terminates the transmission with the proper
       EOT message. The ID string consists of ASCII characters
       followed by carriage return and linefeed. Typically, the
       ID has the following form: two characters for the
       manufacturer code, up to five characters for the model
       number, one model number revision character, and
       optionally other information that describes options or
       other information that the designer feels should be
       included to clarify the identification and capabilities of
       this device. If a device does not implement device ID,
       the SDI message will simply be returned to the controller.

SOT − Start of transmission. Addressed ready group, ready
       class, 101 01100xxx. This subgroup of messages includes
       the SDA, SST, SDI, SAI, and TCT messages. They all serve
       to enable the beginning of transmission from a device
       other than the controller (except for TCT, which enables
       the new controller). The other device will source the
       proper EOT message when finished to signal the controller
       to take over once again. In the case of TCT (take
       control) the new controller remains in control of the loop
       and does not source the EOT. This group of messages does
       not go completely around the loop, but is replaced by the
       first message from the enabled device.

SRQ − Service request. Data or end class or identify class,
       0x1 xxxxxxxx or 111 xxxxxxxx. The SRQ bit is bit C0 of
       data, end, and identify messages. Devices may set this
       bit when they have a need for service from the controller.
       The bit then represents the logical OR of the various
       devices' individual service bits. The controller will
       generally need to perform a serial poll operation to find
       out which device needs service. The command and ready
       classes of messages do not have a service request bit and,
       therefore, do not transmit this message.

SST − Send status. Addressed ready group, ready class,
       101 01100001. The controller uses this message to cause
       the addressed talker to begin sending its status byte(s).
       The talker replaces the SST with its status information
       on the loop. When finished, the talker then transmits the
       proper EOT message. Two bits of the first byte of status
       are reserved for specific purposes. If bit D7 (msb) is
       set, then the first byte of status represents a coded
       system status message. If it is clear, the lower 6 bits
       (D5-0) are device dependent. Bit D6 of the first status
       byte is always equal to the device's local message rsv.
       If the device does not implement serial poll (status), the
       SST frame is simply returned to the controller.

TADn − Talk address n command. Talk address group, command
       class, 100 010aaaaa. The controller uses this command to
       enable one device on the loop to be the active talker.
       The lower five bits (aaaaa) represent the device's address
       (primary address in the case of those devices which
       respond to a two byte address). Address values can range
       from 0 to 30. TAD with address 31 is the UNT (untalk)
       command. If an addressed talker receives UNT or a TAD
       with an address that does not match its own, it must
       become untalked.

TAG - Talk address group. Command class, 100 010xxxxx. This
       group includes all primary talk address commands and the
       untalk command. Secondary addresses are contained in the
       SAG group.

TCT - Take control. Addressed ready group, ready class,
       101 01100100. The active controller uses this message to
       pass control of the loop to another controller. The
       device to which control is passed must first be talk
       addressed, then the current controller sends the TCT
       message. Upon receipt of the TCT, the talker addressed
       device becomes the active controller, and replaces the TCT
       with its first interface message. If the device cannot
       accept control of the loop, it merely retransmits the TCT
       which returns to the current controller, which resumes
       active control of the loop.

UCG - Universal command group. Command class, 100 x001xxxx.
       This group of commands includes all those to which all
       devices respond whether they are currently addressed or
       not.

UNL - Unlisten. Listen address group, command class,
       100 00111111. This command causes all addressed listeners
       to return to the idle state. The controller normally will
       use this command to reset the listeners before addressing
       a new listener(s) for the next data transmission.

UNT - Untalk. Talk address group, command class, 100 01011111.
       This command causes the addressed talker to return to the
       idle state. Since a new talk address causes the previous
       talker to become unaddressed anyway, this command is only
       useful in certain special cases, such as when it is
       necessary to have no talker addressed device on the loop.

)ZES   - Zero extended secondary. Auto address group, ready class,
        101 11000000. This message is used by the controller to
        assign secondary addresses to those devices which have
        multiple address capability. After each device has
        received its primary address via the AMP frame, it waits
        to recognize the ZES frame. When it is received, the low
        order five bits are incremented by the number of addresses
        reserved for this device and the frame is then sent back
        to the controller which now knows how many addresses there
        are in that device. Since the device will only respond
        once to the ZES message, the controller sends it once for
        each device in turn. After this, all devices are
        configured and can respond normally to their assigned
        primary and secondary addresses.

Appendix C. Message Coding

C.1 Command Coding

	x0 0000	x1 0001	x2 0010	x3 0011	x4 0100	x5 0101	x6 0110	x7 0111	x8 1000	x9 1001	xA 1010	xB 1011	xC 1100	xD 1101	xE 1110	xF 1111	
0x 0000	NUL	GTL			SDC	PPD			GET						ELN	ACG	
1x 0001	NOP	LLO			DCL	PPU			EAR							UCG	
2x 0010	LAD 0	LAD 1	LAD 2	LAD 3	LAD 4	LAD 5	LAD 6	LAD 7	LAD 8	LAD 9	LAD 10	LAD 11	LAD 12	LAD 13	LAD 14	LAD 15	LAG
3x 0011	LAD 16	LAD 17	LAD 18	LAD 19	LAD 20	LAD 21	LAD 22	LAD 23	LAD 24	LAD 25	LAD 26	LAD 27	LAD 28	LAD 29	LAD 30	UNL	LAG
4x 0100	TAD 0	TAD 1	TAD 2	TAD 3	TAD 4	TAD 5	TAD 6	TAD 7	TAD 8	TAD 9	TAD 10	TAD 11	TAD 12	TAD 13	TAD 14	TAD 15	TAG
5x 0101	TAD 16	TAD 17	TAD 18	TAD 19	TAD 20	TAD 21	TAD 22	TAD 23	TAD 24	TAD 25	TAD 26	TAD 27	TAD 28	TAD 29	TAD 30	UNT	TAG
6x 0110	SAD 0	SAD 1	SAD 2	SAD 3	SAD 4	SAD 5	SAD 6	SAD 7	SAD 8	SAD 9	SAD 10	SAD 11	SAD 12	SAD 13	SAD 14	SAD 15	SAG
7x 0111	SAD 16	SAD 17	SAD 18	SAD 19	SAD 20	SAD 21	SAD 22	SAD 23	SAD 24	SAD 25	SAD 26	SAD 27	SAD 28	SAD 29	SAD 30		SAG
8x 1000	PPE0 0	PPE0 1	PPE0 2	PPE0 3	PPE0 4	PPE0 5	PPE0 6	PPE0 7	PPE1 0	PPE1 1	PPE1 2	PPE1 3	PPE1 4	PPE1 5	PPE1 6	PPE1 7	ACG
9x 1001	IFC		REN	NRE							AAU	LPD					UCG
Ax 1010	DDL 0	DDL 1	DDL 2	DDL 3	DDL 4	DDL 5	DDL 6	DDL 7	DDL 8	DDL 9	DDL 10	DDL 11	DDL 12	DDL 13	DDL 14	DDL 15	ACG
Bx 1011	DDL 16	DDL 17	DDL 18	DDL 19	DDL 20	DDL 21	DDL 22	DDL 23	DDL 24	DDL 25	DDL 26	DDL 27	DDL 28	DDL 29	DDL 30	DDL 31	ACG
Cx 1100	DDT 0	DDT 1	DDT 2	DDT 3	DDT 4	DDT 5	DDT 6	DDT 7	DDT 8	DDT 9	DDT 10	DDT 11	DDT 12	DDT 13	DDT 14	DDT 15	ACG
Dx 1101	DDT 15	DDT 16	DDT 18	DDT 19	DDT 20	DDT 21	DDT 22	DDT 23	DDT 24	DDT 25	DDT 26	DDT 27	DDT 28	DDT 29	DDT 30	DDT 31	ACG
Ex 1110																	
Fx 1111																	

C.2 Ready Coding

	x0 0000	x1 0001	x2 0010	x3 0011	x4 0100	x5 0101	x6 0110	x7 0111	x8 1000	x9 1001	xA 1010	xB 1011	xC 1100	xD 1101	xE 1110	xF 1111	
0x 0000	RFC																
1x 0001																	
2x 0010																	
3x 0011																	
4x 0100	ETO	ETE	NRD														ARG
5x 0101																	ARG
6x 0110	SDA	SST	SDI	SAI	TCT												ARG
7x 0111																	ARG
8x 1000	AAD 0	AAD 1	AAD 2	AAD 3	AAD 4	AAD 5	AAD 6	AAD 7	AAD 8	AAD 9	AAD 10	AAD 11	AAD 12	AAD 13	AAD 14	AAD 15	AAG
9x 1001	AAD 16	AAD 17	AAD 18	AAD 19	AAD 20	AAD 21	AAD 22	AAD 23	AAD 24	AAD 25	AAD 26	AAD 27	AAD 28	AAD 29	AAD 30	IAA	AAG
Ax 1010	AEP 0	AEP 1	AEP 2	AEP 3	AEP 4	AEP 5	AEP 6	AEP 7	AEP 8	AEP 9	AEP 10	AEP 11	AEP 12	AEP 13	AEP 14	AEP 15	AAG
Bx 1011	AEP 16	AEP 17	AEP 18	AEP 19	AEP 20	AEP 21	AEP 22	AEP 23	AEP 24	AEP 25	AEP 26	AEP 27	AEP 28	AEP 29	AEP 30	IEP	AAG
Cx 1100	AES 0	AES 1	AES 2	AES 3	AES 4	AES 5	AES 6	AES 7	AES 8	AES 9	AES 10	AES 11	AES 12	AES 13	AES 14	AES 15	AAG
Dx 1101	AES 16	AES 17	AES 18	AES 19	AES 20	AES 21	AES 22	AES 23	AES 24	AES 25	AES 26	AES 27	AES 28	AES 29	AES 30	IES	AAG
Ex 1110	AMP 0	AMP 1	AMP 2	AMP 3	AMP 4	AMP 5	AMP 6	AMP 7	AMP 8	AMP 9	AMP 10	AMP 11	AMP 12	AMP 13	AMP 14	AMP 15	AAG
Fx 1111	AMP 16	AMP 17	AMP 18	AMP 19	AMP 20	AMP 21	AMP 22	AMP 23	AMP 24	AMP 25	AMP 26	AMP 27	AMP 28	AMP 29	AMP 30	IMP	AAG

C.3 Frame Hierarchy

```
DOE .... DAB                           IDY .... IDY
         . DAB(SRQ)                           . IDY(SRQ)
         . END
         . END(SRQ)                    RDY .... RFC
                                              .
CMD .... ACG .... NUL                         . ARG .... EOT .... ETO
         .       . GTL                        .       .        . ETE
         .       . SDC                        .       .
         .       . PPD                        .       . NRD
         .       . GET                        .       .
         .       . ELN                        .       . SOT .... SDA
         .       . PPE0(0-7)                   .       .        . SST
         .       . PPE1(0-7)                   .       .        . SDI
         .       . DDL(0-31)                   .       .        . SAI
         .       . DDT(0-31)                   .       .        . TCT
         .                                    .
         . UCG .... NOP                        .
         .       . LLO                         . AAG .... AAD(0-30)
         .       . DCL                                 . NAA(1-31)
         .       . PPU                                 . IAA
         .       . EAR                                 .
         .       . IFC                                 . AEP(0-30)
         .       . REN                                 . IEP
         .       . NRE                                 .
         .       . AAU                                 . ZES
         .       . LPD                                 . AES(0-30)
         .                                             . NES(1-31)
         . LAG .... LAD(0-30)                          . IES
         .       . MLA(0-30)                           .
         .       . UNL                                 . AMP(0-31)
         .                                             . NMP(1-31)
         . TAG .... TAD(0-30)                          . IMP
         .       . MTA(0-30)
         .       . OTA(0-30)
         .       . UNT
         .
         . SAG .... SAD(0-30)
                  . MSA(0-30)
                  . OSA(0-30)
```

C.4 Accessory Identification

        Accessory ID provides HP-IL Controllers with the ability
to quickly identify the devices on the loop according to device
functions.   The accessory ID consists of a four bit class
descriptor, and a four bit type field.   The class descriptor
indicates what main function the device provides such as printer,
mass storage, etc.   The type field indicates specific attributes
about the device.   If a device exhibits similar characteristics
as other devices within a particular class and type, the device
should respond to accessory ID with that class and type.   If no
type exists within a device's class that closely matches its
attributes, the device should respond with type E of that class.
The extended class (Fx) and extended type (xF) are reserved to
allow new classes and types.

        It is very strongly recommended that all future devices
designed for HP-IL systems respond to accessory ID.   HP-IL
controllers should be able to expect the accessory ID to be
available so that certain system I/O functions can be executed
automatically.

Classes and Types Defined (hex):

Øx Controllers

    ØØ Limited controller capability mostly automatic
       system I/O functions.
            example device: HP-41C
    Ø1 Full instrumentation controller; completely manual
       (programatic) control.
            example device: HP Series 8Ø
    Ø2 Full interface controller completely automatic
       including control passing during I/O operations
    Ø3 Full interface controller partially automatic

    ØE general controller

    ØF extended (reserved)

1x Mass storage devices

      10 Seek/read/write protocol using device dependent commands
         as defined by the HP 82161A Digital Cassette Drive.
               example device: HP 82161A Digital Cassette Drive
      1E general mass storage

      1F extended (reserved)

2x Printers

      20 24 column; HP escape sequences; column dot graphics
               example device: HP 82162A Thermal Printer
      21 80 column; HP escape sequences; column dot graphics
               example device: HP 82905B Impact Printer
      22 80 column; HP escape sequences with no graphics
               example device: HP 2671A Thermal Printer
      23 80 column; HP escape sequences; HP raster graphics
               example device: HP 2671G Thermal Printer
      24 80 column; HP escape sequences; HP raster graphics
               example device: HP 2673A Thermal Printer
      2E general printer

      2F extended (reserved)

3x Displays

      30 32 column; HP escape sequences; no graphics
               example device: HP 82163A Video Interface
      3E general display

      3F extended (reserved)

4x Interfaces

      40 HP-IL/GPIO interface
               example devices: HP 82165A, HP 82166A
      41 HP-IL modem
               example device: HP 82168A
      42 HP-IL/RS-232-C interface
               example device: HP 82164A
      43 HP-IL/HP-IB interface
               example device: HP 82169A
      4E general interface
      4F extended (reserved)

)5x   Electronic Instrumentation.

      51-57   If bit D3 is Ø the lower three bits (D2-DØ)
         represent the functions contained in this device.
         D2, D1 and DØ are defined as signal source, signal
         switch and signal measurement functions respectively.
         For example a device that can act as both a signal
         source and measurement device would respond 55.

      5E general electronic instrument

      5F extended (reserved)

6x   Graphic I/O

      6Ø HP-GL compatible
               example device: HP 747ØA Plotter

      6E general graphic I/O device

      6F extended (reserved)

7x   Analytical and Scientific Instrumentation

      7E general analytical or scientific instrument

      7F extended (reserved)

Ex   General devices. This class consists of devices that do not
      easily fit into the other classes listed.

      EØ EPROM programmer

      EE general

      EF extended (reserved)

Fx   Extended class (usage not currently defined)

C.5   System Status Messages

HP-IL has the capability for a device to report status to the controller in an interface defined code called system status. It is required that all devices on HP-IL reserve the most significant bit of the first byte of status to indicate system status. Bit 7 (msb) is required to be zero when the device sources device dependent status, and one when the device sources system status. More than one byte of status may be sent, but all bytes after the first one are designer specified.

It is very strongly recommended that devices use system status messages whenever possible. This allows generalized controllers to be implemented that can understand status messages from any device on the loop. Most devices can describe all possible status with the system messages defined. If more specific information is necessary or desirable, the device may send additional bytes of status.

Generally, there two types of status messages. State messages represent the overall status of the device. A state message will remain true within a device until another message of higher priority replaces it. All OK is an example of a state status message. Event messages indicate an occurrence within a device that does not necessarily affect the current state of the device. Once the controller has been informed of the situation through a serial poll, the device removes the event system status message. Data Error is an example of an event status message.

Except for the All OK message, the event and state system status messages can be distinguished by the value of bit D5. If bit D5 is set, the message is a state or condition. If bit D5 is clear, the message is an event. The all OK message is a state, but the value of this message is all zeros.

Devices must select the most important system status message to report during each serial poll. The most important status message is always the one having the highest priority that is currently true within the device. The following table lists all system status messages and their meanings in priority order with the highest priority messages appearing first.

System Status Messages

Device Events

```
DDDDDD
543210    Message definition
```

000110   Self test failure. The device has discovered a
          condition that makes proper operation impossible to
          guarantee. Highest priority of all system status
          messages.

001000   Powering Down. The device indicates to the controller
          with this message that it is about to power down and
          therefore, the loop will become inoperative shortly.
          The message is only useful if the device can delay
          powering down long enough to report the condition.
          Note that this message is not to be used if the device
          is powering down in response to an LPD command.

001001   External Service Request. With this message, the device
          indicates to the controller that an external input (such
          as a switch) has been activated.

000010   Manual intervention required. The device cannot
          function as designed until an action has been performed
          by the operator.

000011   Data error. The device has detected a situation that
          caused information to be lost in a device dependent
          manner not associated with the talker function. If data
          is lost due to a loop transmission error, the End Of
          Transmission with Error (ETE) ready frame will be
          sourced by the talker.

000111   Command Error. This message indicates to the controller
          that an invalid device command was received.

000101   No room. The device does not have sufficient space for
          the data being received from the loop and the situation
          will not be resolved without intervention.

000100   Device error. The device is in an invalid state or
          cannot perform the desired operation. This status may
          be the result of an incorrect sequence or combination of
          device commands or may be due to an error condition
          within the device.

System Status Messages (continued)

000001   Low battery. The device has detected that the battery
power has reached a critical level. Future performance
of the loop is endangered if this message is ignored.

001010   Device Dependent Service Request. The device has
detected an event which requires attention from the
controller. This message should be used only if no
other system status message is appropriate.

011111   ASCII Follows. The bytes following this message (until
an ETO) represent an ASCII message for the user or
operator. Lowest priority of device event messages.

Device States

```
DDDDDD
543210   Message definition
```

100000   Request control of loop. A device may indicate to the
current controller with this message that it has need
for control of the loop. Highest priority of device
state messages (lower priority than event messages).

100010   Ready to send data. The device has data available and
is currently ready to source it on the loop.

100001   Ready to receive data. The device indicates with this
message that it is currently ready to accept data from
the loop.

100011   Not ready to receive or send data. The device indicates
to the Controller with this message that it has no data
ready to send and that it is not able or ready to accept
data. Without any intervention, the device will
eventually correct the situation and become ready.

000000   All OK. This message indicates that the device has no
need for attention and is currently ready to perform the
operations for which it was designed. Lowest priority
of all system status messages.